Joint Master's Degree in Sustainable Automotive Engineering

Master Thesis

Temporal Graph Convolutional Deep Neural Networks for Traffic Forecasting using GPS Data

Toon Bogaerts

Academic & Industrial Supervisor: Enrique Onieva Caracuel

Bilbao, 10/06/2019

Declaration

I understand that all my project work must be my own work. If I make use of material from any other source I must clearly identify it as such in any interviews, reports or examinations. I understand that my reports must be written unaided in my own words, apart from any quoted material which I must identify clearly in the correct manner.

I understand that the work which I shall present for assessment must be work carried out by myself only during the project period which has not been previously prepared. Where any such previous work is made use of in the project, I shall make this clear in any interviews, reports or examinations.

I understand that violation of these conditions may result in a mark of zero for the whole subject, the component or components of assessed work affected.

Student name: Toon Bogaerts

DNI/ID Card/Passport: EP388057

Signature

Date: 10/06/2019

Universiteit Antwerpen université de BORDEAUX Deusto Universidad de Deusto University of Deusto Loughborough University

Abstract

Traffic forecasting is an important research area in Intelligent Transportation Systems as being able to anticipate traffic is one major step to solving congestion. Deep neural networks have proven excellent results in different predictive applications. In this work, a deep neural network is proposed, which simultaneously extracts spatial features of traffic using graph convolution and temporal features with Long Short-Term Memory (LSTM) cells to make both short-term and long-term predictions. These predictions are made for a single link and on a network-wide scale. The models are trained and tested using sparse GPS-data coming from the ride-hailing service of DiDi in the cities of Xi'an and Chengdu in China. Besides presenting the deep neural network, a time-related data reduction technique is proposed to select the most relevant road links to be used as input for a single link or on a network-wide scale. The sigmoid activation function is introduced on the output layer for predicting traffic speed to improve the training process. Combining the suggested approaches, the proposed models achieve better performance compared to high-performance algorithms for traffic forecasting such as LSTM and advanced algorithms used in the TRANSFOR19 forecasting competition. A first submission was ranked 3rd in this world-wide forecasting competition. In this work, these models are streamlined and have increased accuracy, performance and are applied to more complex problems. The network-wide predictions score averagely better if compared to the best performing algorithm used in the forecasting competition, which only predicts for two links. The single link model is capable of maintaining its performance over different time-horizons ranging from five minutes up to four hours with multistage predictions. This time-horizon is currently limited to one hour in multi-step for the network-wide predictions.

Keywords:

Deep Learning, Supervised Learning, Machine Learning, Traffic Forecasting, GPS-Data, Intelligent Transportation Systems

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

CMCM	Convolution-Maxpooling-Convolution-Maxpooling
CNN	Convolution Neural Network
DL	Deep Learning
DNN	Deep neural network
GRU	Gated Recurrent Unit
ITS	Intelligent Transportation System
K-NN	k-nearest neighbours
LSTM	Long Short Term Memory
MAE	Mean-Absolute-Error
MAPE	Mean-Absolute-Percentage-Error
MLP	MultiLayer Perceptron
MMCMC	Maxpooling-Maxpooling-Convolution-Maxpooling-Convolution
MSE	Mean-Square-Error
RMSE	Root-Mean-Square-Error
RNN	Recurrent neural network
SVR	Support Vector Regression
TF	Traffic Forecasting

170

Acknowledgement

Firstly, I would like to thank my thesis supervisor Dr Enrique Onieva Caracuel. He consistently allowed me to conduct my research but steered me in the right direction whenever he thought it was needed. I could always visit his office whenever I had any doubts about my research or writing.

I would also like to thank Dr Antonio D. Masegosa. He was always available to give valuable feedback on every idea I proposed. We formed a great team in solving complex problems. Without his input, the quality of research would have suffered.

Next, I would like to acknowledge Juan S. Angarita-Zapata. Together with Antonio and Enrique, they co-authored my research paper. Without their knowledge and experience, it would not have been possible to publish the paper.

As student representative, I was in close contact with the steering committee of this international joint master. I want to thank the steering committee for their contribution to this program, which allowed students to receive numerous valuable opportunities which are not common in other programs.

Finally, I must express my gratitude to my parents and especially to my girlfriend for providing me with support and continuous encouragement throughout my two-year study abroad. This accomplishment would not have been possible without them. Thank you.

1 Introduction

The study of Traffic Forecasting (TF) has become an important aspect of Intelligent Transportation Systems (ITSs). TF has many contributions to traffic flows with the capability of preventing congestion. As congestion mostly appears in cities, TF is applied in densely populated areas. The main objective of TF is the prediction of traffic measures in the near future, ranging from the next few minutes up to several hours, based on historical traffic data [1]. A standard procedure for TF has been based on traffic theory models and classical statistical methods [2], [3]. At present, the data available from ITS has caused a shift from these approaches to data-driven models. Within data-driven approaches, statistical and Machine Learning methods are the two main categories. In the early stages of the application of data-driven methods to TF, a broad literature on statistical approaches can be found, such as ARIMA [4]. However, they have shown shortcomings to deal with high-dimensional and complex TF problems. For this reason, most of the current literature in this area is focused on Machine Learning methods. These models have better capabilities to address high-dimensional data and extracting non-linear relationships.

Some well-known examples of traditional Machine Learning methods applied to TF are Support Vector Regression (SVR), K-Nearest neighbours (k-NN) or Random Forest, among others [3], [5]. Despite their success, classical Machine Learning methods present limitations when capturing the spatial and temporal relationships that are usually observed in traffic patterns [6]–[8]. In recent years, Deep Neural Networks (DNNs) have caught the attention of transportation research [9], [10]. This increase is due to their high representational power of the features present in traffic. This combined with their high performance in terms of accuracy and error metrics [11]. Some early examples of Deep Learning (DL) applied to TF are Deep Belief Networks [12] and stacked auto-encoders [13]. More recently, they shifted to the use of other types of DNNs such as Recurrent Neural Networks (RNNs) [14], [15] or Convolutional Neural Networks (CNN) [16], [17].

The main reason behind the use of RNNs is their ability to capture short and long temporal dependencies, especially Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). In the case of CNN's, the main motivation behind their application is the capacity to model spatial relationships. Given that conventional CNN's were not designed to deal with graph-structured data, as it is the case of traffic information and road networks, researchers developed variants to capture better spatial relationships that are commonly known as Graph CNN [18], [19]. In last years, hybrid architectures emerged that combine RNNs and Graph CNN with the aim of capturing simultaneously the traffic spatial and temporal relationships; these approaches have demonstrated superior performance than their ”pure” RNN and Graph CNN counterparts [14], [7], [14], [15], [20].

On the one hand, as far as known, previous works on hybrid GraphCNN-RNN architectures were designed to work with point-wise traffic sensors or a reduced set of road links for limiting the complexity and dimension of the input data. On the other hand, they are only tested for short-term predictions (time horizon <60 minutes). Regarding the first issue, it is especially relevant in the case of floating car data. Traffic information is usually spread around the whole city, and the temporal dependencies between road links are not obvious nor confined to a reduced area around the targeted segment. As for the second issue, this is not restricted to GraphCNN-RNN methods, but it is a common gap in TF literature [5]. However, it is interesting to analyse if the promising results that they achieve for short-term predictions can be extrapolated to long-term horizons.

According to the state-of-the-art literature, the current general and computational challenged of DL methods applied to TF are:

- The scalability of DL methods for large-scale network predictions [14].
- The efficiency of training strategies for models that can be applied in real-time applications [21].
- Including the influence of non-recurrent events such as accidents or special events [7].
- The design of hybrid models which are capable of evaluating both spatial and temporal relations in traffic [22].
- Long-term predictions on a network level [23].
- The construction of DL methods which can deal with missing and erroneous data, as well as the randomness in the temporal-spatial coverage of GPS-data.

The objective of this thesis is to develop a novel GraphCNN-RNN architecture for high precision TF based on GPS-data. The architecture will be designed to predict for long-term horizons on both a local and network-wide scale. The model should manage both spatial and temporal relations within the traffic data and use this information for improving predictions. Furthermore, an efficient data pre-processing technique should be created to handle today's current challenges of graph CNN-RNN architectures.

This thesis is related to my paper, which is written for a special issue in the "Transportation Research Part C" journal. This journal has an impact factor of 3.968[1]. Currently, the paper is in review and will most likely be published in September 2019.

An initial version of this model was presented in TRANSFOR19 competition, ranking 3rd[2]. The Transportation Forecasting Competition (TRANSFOR19) was organized by the Standing Committee on Artificial Intelligence, the Advanced Computing Applications (ABJ70) of the Transportation Research Board, and the IEEE ITS Technical Activities Sub-Committee "Smart Cities 70 and Smart Mobility". As part of the competition, I was invited to present our work in Washington DC during the TRB-annual meeting 2019 in January.

The thesis is structured as followed: Section 2 presents the related work in the area of DL approaches applied to TF. Section 3 summarises the experimental set-up, followed by the results in section 4. Section 5 discusses the results obtained; Section 6 summarises the thesis. Lastly, Section 7 suggests future work.

[1] https://www.journals.elsevier.com/transportation-research-part-c-emerging-technologies
[2] https://sites.google.com/site/trbcommitteeabj70/abj70s-latest-news/another-successful-trb-for-abj70?authuser=0

2 Literature review

This section reviews the literature related to modelling and prediction of traffic using ML approaches. Section 2.1 will focus on statistical and ML approaches to TF. Section 2.2 summarises and discusses related works that have been presented using DL methods applied to TF. To finish this chapter, a review of current commercial products for traffic predictions will be discussed in section 2.3.

2.1 Data-driven modelling approaches for Traffic Forecasting

TF can be approached from different modelling perspectives. The four most common approaches found in transportation literature are:

(i) The statistical time-series perspective [9]

(ii) The supervised regression problem [24]

(iii) The supervised classification problem [25]

(iv) Clustering-pattern recognition approach [26]

First, the time-series perspective is based on developing models that fit observations made at prior times and using them to predict future traffic [27]. This is at the expense of being able to explain why a specific prediction was made and understanding the underlying causes behind the TF problem at hand [28]. Secondly, the supervised regression approach is focused on building a predictive model without prior models or error distribution specifications. This while using historical data to make predictions of typically continuous traffic measures based on unseen data [9].

Third, the supervised classification approach focuses on forecasting a discrete traffic value based on historical data (continuous traffic measures such as flow or speed) [16][29]. It is important to clarify that the forecasting of discrete variables could also be addressed as a regression problem predicting either speed or density (continuous values), followed by categorising these predictions to obtain discrete outputs. Lastly, the fourth modelling approach is the clustering-pattern recognition problem. Its focus is on investigating the dynamic traffic relationships of different locations. This is achieved by characterising similar traffic measures values from one road to another and then grouping the locations in clusters that divide the network into correlated groups [30]. This is possible due to traffic's correlations in the temporal and spatial domain. Exploiting such similarities enables traffic conditions to be predicted cluster by cluster for future times, based on historical traffic data.

Within this work, the focus is on predicting traffic using a supervised regression approach. Concretely, DL methods which can forecast future states of traffic are proposed and tested. The output will be continuous, and the models are trained on historical traffic GPS-data. In the following subsection, related works that utilise DL methods to predict traffic under the same modelling perspective are presented.

2.2 Deep Learning approaches for Traffic Forecasting

The literature on TF reports a wide variety of ML methods to predict traffic such as Neural Networks, Support Vector Machines, k-Nearest Neighbours or Random Forest. These ML methods have attracted more attention in recent years as more evidence has become available about how they can outperform statistical methods in their prediction of traffic regarding different scenarios. Despite these achievements, there are still open issues concerning more complex ML approaches that forecast traffic using GPS-data. According to the state-of-the-art findings, DL methods predicting the mentioned data source have demonstrated superiority in terms of accuracy and computational performance compared to statistical approaches and traditional ML methods

Reviewing recent literature, one of the first contributions of DL to forecast traffic can be dated from 2015 when [31] proposed a deep NN that considers the spatial and temporal correlations of traffic data to make predictions. Although the input data used was not captured by GPS sensors, and therefore the forecasting of traffic was not focused on the network level. This work was the predecessor of more recent contributions in the same area. Three years later, [15] implemented a traditional Convolutional NNs (CNN) architecture able to convert traffic speed data into a series of static images from which traffic predictions are made. Despite the novelty of considering CNN's for TF, the image representation of the traffic data sometimes could not capture the real physical characteristics of the network under study. Which in turn, leads to making predictions that do not correspond to the actual traffic conditions on the roads.

Later in 2018, [14] proposed a deep architecture of a Long Short-term Memory (LSTM) NN that captures spatial features of traffic data in small freeway and urban environments. In a sense, the authors open the path to explore the potential of LSTM NNs to approach TF in larger traffic networks. This network can include the full range of spatial dependencies among all road segments. Furthermore, the authors discuss possible enhancements of the performance of the proposed model through the incorporation of non-traffic data into DL models. In concordance with this last issue, [32] developed a DL architecture that models the nonlinear spatiotemporal effects in recurrent and non-recurrent traffic congestion patterns. These patterns include data about construction zones, weather, special events and traffic incidents. However, the model presented by the authors was not able to forecast traffic at the network level due to the use of fixed position sensors, which suffer from spatial coverage issues.

Improvements were made by [14] and [32], [21] and [22] developing Graph Convolutional LSTM NNs to learn the interactions between roadways in a network and then forecast the network-wide traffic state. These were their main contributions to the state-of-the-art. Although previous DL-based methods can learn such spatial dependencies, they tend to be over-complex and inevitably capture a certain amount of noise and spurious relations. These relations are not likely to represent the true physical structure in a traffic network. Then, [33] presented a hybrid NN to predict urban flow that combines a CNN to statically extract the spatial features of traffic data with an LSTM to extract the temporal dimension of traffic. Concerning the work done by [21] and [22], [33] presented a model that can perform long-term prediction, which is valuable for improving the management of traffic at the network level.

More recently, [34] added non-traffic data to the input layer of the DL architecture proposed. The main contribution of this model is its ability to reducing the accuracy degradation caused by missing data. Nevertheless, the spatial coverage of predictions is focused on a single road segment, which limits its practical application in real transportation scenarios. Later, [7] proposed a novel deep NN that mixes a temporal Graph CNN component with a gated recurrent unit (GRU). As proposed by [21] and [22], the graph-based model architecture is used to capture the topological structure of the road network and to model the spatial dependence between different locations. The suitability of using graph-based approaches and CNN's to approach TF is promising and well documented in the transportation literature. In this context, more recent challenges are centred in proposing DL methods that allow exploiting the benefits of GPS data, such as is stated by [23]. In this works, the authors propose a spatiotemporal LSTM NN presided by map-matching named as MapLSTM. The purpose of the method is predicting fine-grained traffic conditions based on moving sensors data. First, the MapLSTM obtains the historical and real-time traffic conditions of road segments via map-matching. Then the LSTM component is used to predict the conditions of the corresponding road segments in the future.

Summarized, DL methods have demonstrated to be capable of dealing with the nonlinear dependencies of traffic data while maintaining high performance in terms of accuracy. However, DL methods and its applications can be further improved to make them more flexible and effective in real transportation scenarios.

2.3 Commercial product for traffic predictions

At present, there are several commercial products for TF. Most known are the services of Google being Google Maps, and Waze showed in Figure 1. Google Maps estimates the travel time based on historical data combined with a routing algorithm to optimise travel time or distance. Users can use this feature for most modes of transport. Google collects enormous amounts of data from their users, providing them with real-time updates, more accurate predictions, ….

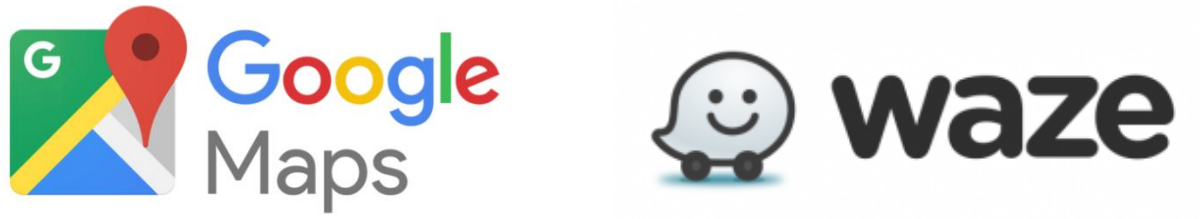


Figure 1: Google's commercial TF application

Waze focusses more on cars compared to Google Maps. The main difference is that Waze is community driven while Google Maps is more data-driven. Waze relies on driver's collected data to update their maps in real-time. To have these real-time updates, an internet connection is needed while Google maps can work offline. If you plan a route in the future, Google Maps calculates the most likely duration range of your trip. How this duration range is calculated is sadly not shared by Google.

Nokia's HERE WeGo is another routing application providing traffic information shown in Figure 2. In general, it has the same functionality as Google's apps but falls a bit short on services like dynamic route updates.

Figure 2: HERE WeGo

These applications provide real-time information and estimate future travel times based on historical data. If additional predicted data is used, these applications might be able to prevent congestion by redirecting their users based on the predicted values. The current trend of increasing connectivity will result in more data, allowing for more advanced algorithms providing accurate predictions.

3 Methodology

This section will describe the steps taken from raw data to predictions. Section 3.1 will be devoted to different strategies used in the pre-processing of data. Section 3.2 will focus on the technical build-up of the hybrid neural network architecture. Section 3.3 will end this chapter with the experimental setup.

3.1 Data pre-processing

This section will be devoted to explaining the process of handling GPS data to create reliable data to forecast traffic regarding average speed.

3.1.1 Data description

The proposed architectures have been trained and tested on GPS-data from The GAIA initiative by DiDi. This data set contains trajectory data of DiDi Express and DiDi Premier drivers within the cities of Xi'an and Chengdu in China. The data contains trips from October and November 2016. The sample period of the measured GPS track points is approximately two to four seconds. The order and trip information are anonymised, Table 1 shows an overview of the dataset. The total size of the dataset in CSV format is 36.5 GB in a comma-separated format. This dataset was used in the TRANSFOR 2019 traffic forecasting competition. As most GPS data sets, this set contains only a specific part of the traffic population but will be considered as ground truth. Compared to induction sensors which contain the full population of vehicles crossed, GPS-data has better coverage of the city. Induction sensors have fixed locations and are, in most cases, only deployed on major highways. In ideal scenarios, all vehicles would be traced with GPS to create the full population. Sadly, most datasets are private and are not shared by the companies collecting them.

Field	Sample	Comment
Driver ID	glox.jrrlltBMvCh8nxqktdr2dtopmlH	Anonymized
Order ID	jkkt8kxniovIFuns9qrrlvst@iqnpkwz	Anonymized
Time Stamp	1501584540	Unix timestamp, in seconds
Longitude	104.04392	GCJ-02 Coordinate System
Latitude	104.04392	GCJ-02 Coordinate System

Table 1: Route data description

From the two months of data, the first seven weeks will be used as training and the last week as a test set. With time-dependent data, it makes the most sense to split data based on time so algorithms do not see information which would not be available if the data set is run in sequence. For all pre-processing and training, the test set will not be used. Traffic flow has a weekly trend, as shown in Figure 3, so using one week as test set makes sure the algorithms do not overfit one particular day.

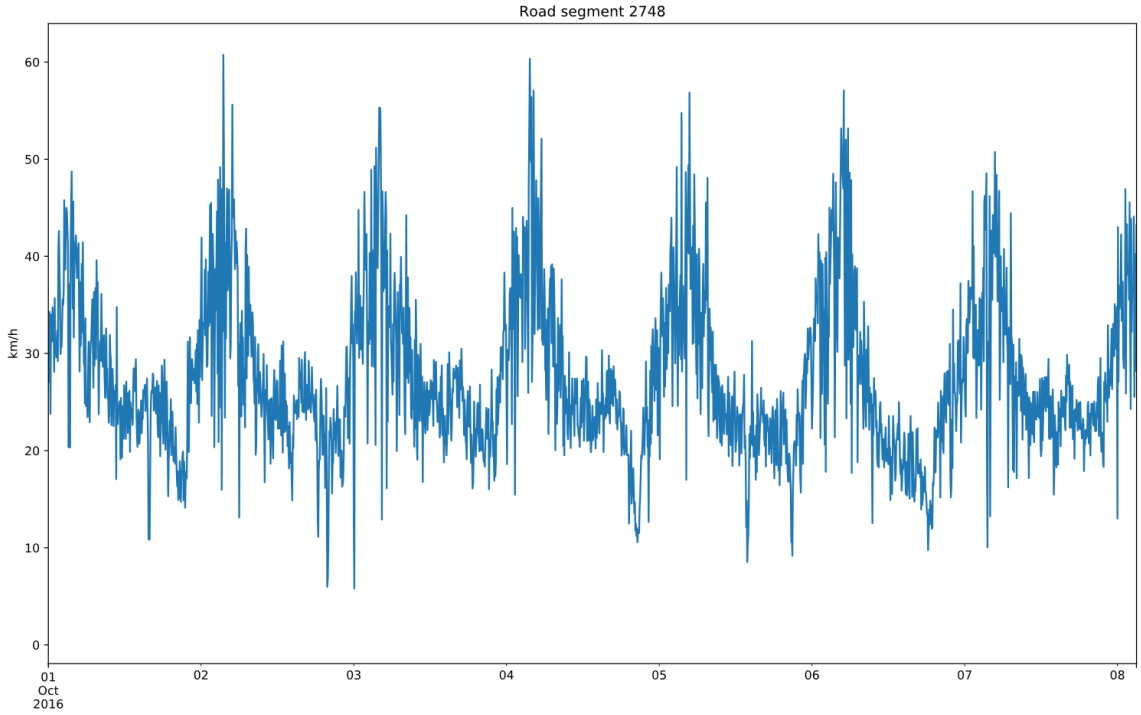


Figure 3: Aggregated average speed during the first week of October for Xi'an segment 2748

3.1.2 Map matching

The map-matching process consists of finding the most probable sequence of road links followed by a vehicle, given the GPS trace of that vehicle that is usually noisy and inaccurate. In this work, the approach proposed by Newson and Krumm in 2009 [35] based on Hidden Markov Models are used. In this method, the observations correspond to the GPS measurements, and the probability that a GPS measurement corresponds to a specific link is inverse to the distance to that road link. This is calculated using a Gaussian function. The Viterbi algorithm [36] was used to find the most probable sequence in the Hidden Markov Model that, in turn, corresponds to the most probable sequence of road links for that GPS trace. The basic process of map matching is shown in Figure 4. Because of the high sampling rate of GPS-data (2-4s), the map-matching algorithm makes almost no errors.

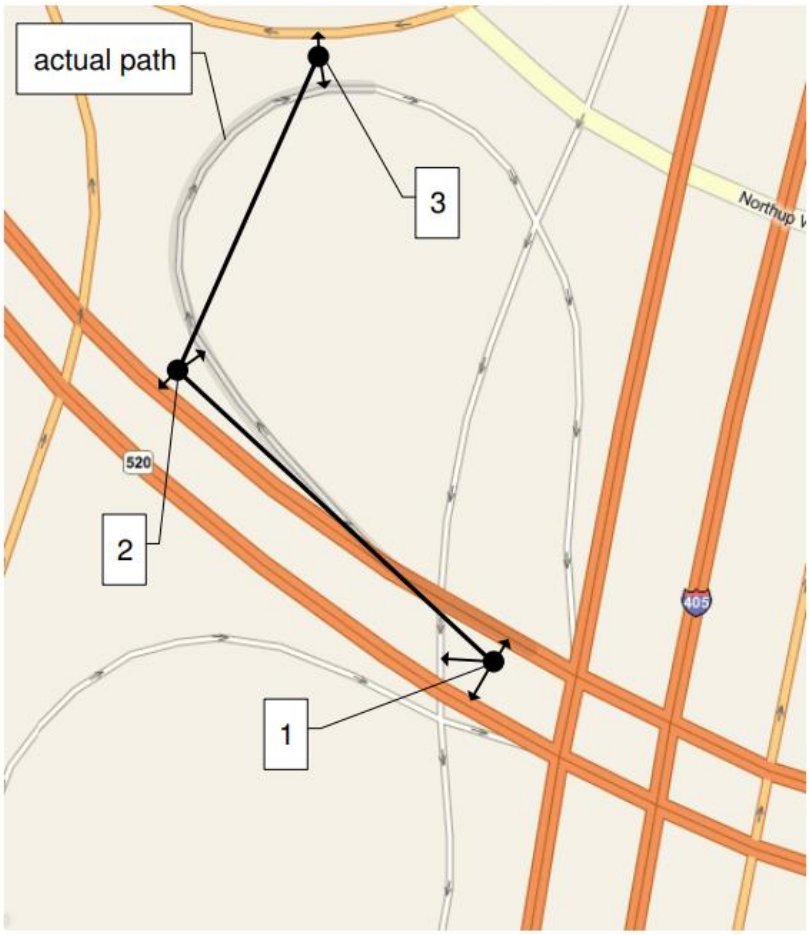


Figure 4: Illustration of map-matching, source [35]

3.1.3 Aggregation

Following the map matching an aggregation is performed to group data, to reduce its size and the computational resources needed to process it. Individual entities of map matched data contain information of the averaged speed in the corresponding edge. These entities are then grouped by a time interval of five minutes. The aggregation will take the average speed within these time intervals and count the number of measured cars. This approach gives a fixed structure to the data, as shown in Figure 5.

Time	Average speed [km/h]	Number of cars [units]
8:00	60	12
8:05	56	15

$$\mathbf{5\ min}\ \boldsymbol{in}\ \mathbf{2}\ \boldsymbol{months} = \mathbf{17857}$$
$$\boldsymbol{edges\ in\ Xian} = \mathbf{5505}$$
$$\boldsymbol{features\ per\ edge} = \mathbf{2}$$
$$\mathbf{dim}\ \boldsymbol{data} = (\mathbf{5505}, \mathbf{2}, \mathbf{17857})$$

Figure 5: Data aggregation example

Where data is missing, the algorithm will implement the value of NAN (Not Available Number), so these gaps can be identified by the following algorithms.

3.1.4 Missing values

With the use of GPS-data coming from ride-hauling services, missing data is a common issue. As the number of requests is lower during the night, the available data during these hours is sparse. This issue has more impact on smaller roads as they already have a lesser amount of measurements. The missing data is filled with two different approaches. For small gaps of maximum two consecutive time steps, linear interpolation among the previous and next step is used. In case the gap of information is larger than two measures, an average based imputation method is used. This average is calculated on the available training data.

Figure 6 illustrates the strategy based on a fictive example. At first, the average is calculated on the values within the same five-minute interval and weekday. In case these values are missing as well, the criteria are reduced to the same time stamp independent of the weekday, next, the same quarter of the hour is used. Finally, in case everything else fails, the same hour and day are used for averaging. Figure 7 shows an example of a segment with about 10% missing data. After these steps, the data has a fixed size and contains no more missing data in each one of the road segments.

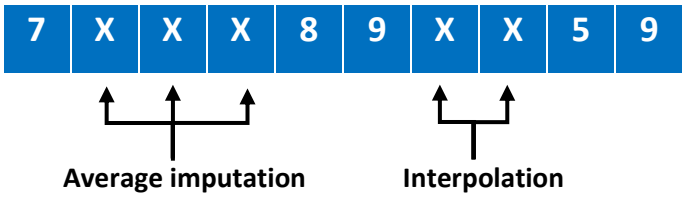


Figure 6: Missing values solution

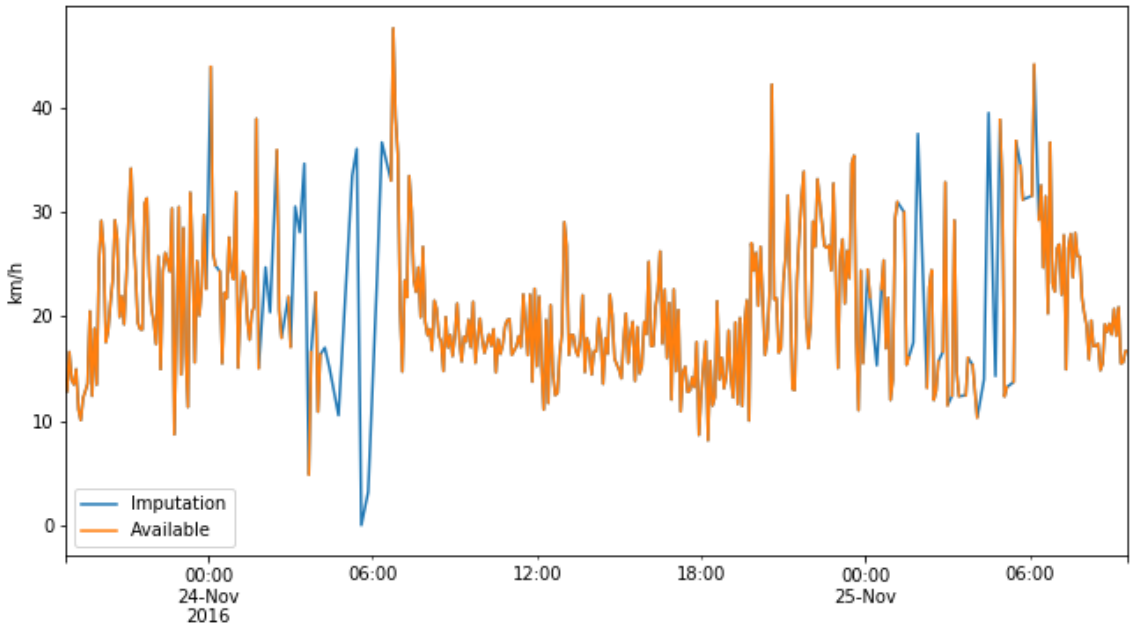


Figure 7: Example of imputation on segment 7210 in Xian

After filling the missing values, a statistical analysis is performed. This analysis gives a deeper insight into the content of the data set. The analysis of some segments is shown in Table 2. A difference is made between the full analysis and a windowed one. The window is set, so it contains morning and evening rush-hours. These events have a high influence on traffic flow and thus are regions of interest.

Edge ID	mean	std	min	25%	50%	75%	max
36077	22,16	6,27	0,00	18,12	20,96	25,67	60,26
463	37,20	9,52	0,00	30,84	36,18	44,89	69,69
9	21,73	4,49	0,00	19,18	21,38	24,23	51,23
36000	21,87	6,55	0,00	17,32	20,65	25,86	83,06
Edge ID	**win_mean**	**win_std**	**Win_min**	**Win_25%**	**Win_50%**	**Win_75%**	**win_max**
36077	19,52	4,92	0,00	16,73	19,18	21,77	47,04
463	33,73	8,00	0,00	29,04	33,78	37,95	61,74
9	20,55	3,95	0,00	18,32	20,41	22,60	46,21
36000	18,99	4,92	0,00	15,81	18,48	21,56	51,04

Edge ID	measurements	intensity	missing
36077	133607	47,42%	21,76%
463	199999	70,99%	17,46%
9	112654	39,99%	27,36%
36000	90135	31,99%	25,11%

Table 2 Example of statistical analysis, Win: windowed implementation

The intensity column represents the percentage of measurements compared to the maximum measured in all segments. The last column contains the missing ratio before missing value imputation. From now, the primary data structure is complete. This data structure is saved to be fixed so other algorithms cannot change any data. The fixed structure contributes to the robustness of the program as different experiments will use this data.

3.1.5 Data reduction

Data reduction is of great importance to alleviate the computational resources needed to process large data sets. The reduction speeds up the training stage of the models, but also simplifies the network architecture and reduce the required hardware. Performing data reduction always implies a trade-off between the advantages mentioned above and the loss of information. When exploring urban traffic data, it can be found that numerous roads contain similar information. Think of a traffic jam on an intersection; the congestion can be identified on all connecting roads. Considering these ideas, the data set will be reduced with the following filters.

- Hierarchal road selection
- Graph simplification
- Correlation
- Time-based correlation
- Traffic flow in-out

Filter combinations are evaluated on performance, and the best filter combination will be used for further research. Following sections will explain each filter in more detail.

3.1.5.1 Hierarchal road selection

Regarding TF, we can solely focus on the most influential roads as inputs. These roads can give an image of the traffic situation over the entire city. One reason why smaller roads are eliminated is the use of sparse GPS data. Most smaller roads have high percentages of missing data resulting in unreliable information. Even though some streets have enough measurements, similar information is mostly contained in connecting primary roads or clusters. The classification used during this selection is created by OpenStreetMap while map-matching, as explained in section 3.1.2. The reduction is shown in Figure 9. In Figure 9 A, you can see all nodes and edges extracted during the map matching. Figure 9 B illustrates primary, secondary and tertiary roads. Lastly, Figure 9 C illustrates only primary and secondary roads; this network will be used in further steps. Figure 8 illustrates the difference between the three networks in a small area. The justification of this reduction comes from a small experiment with the conclusion that the primary and secondary roads still contain enough information to make accurate predictions.

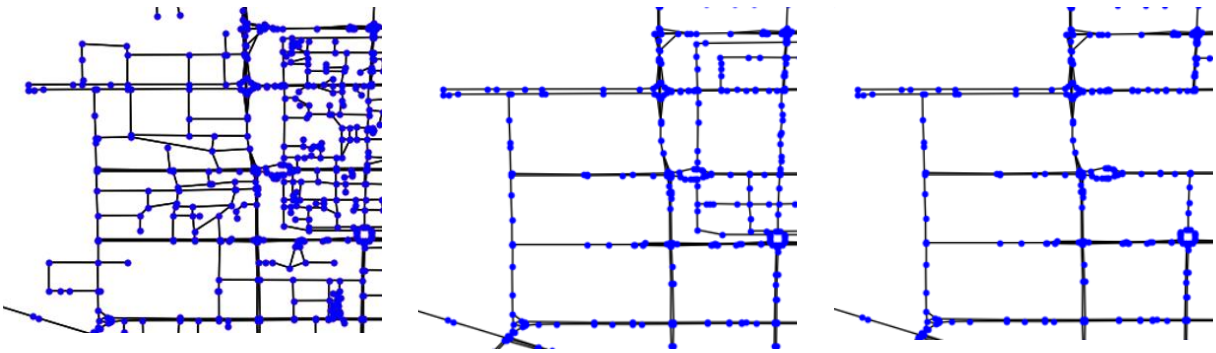
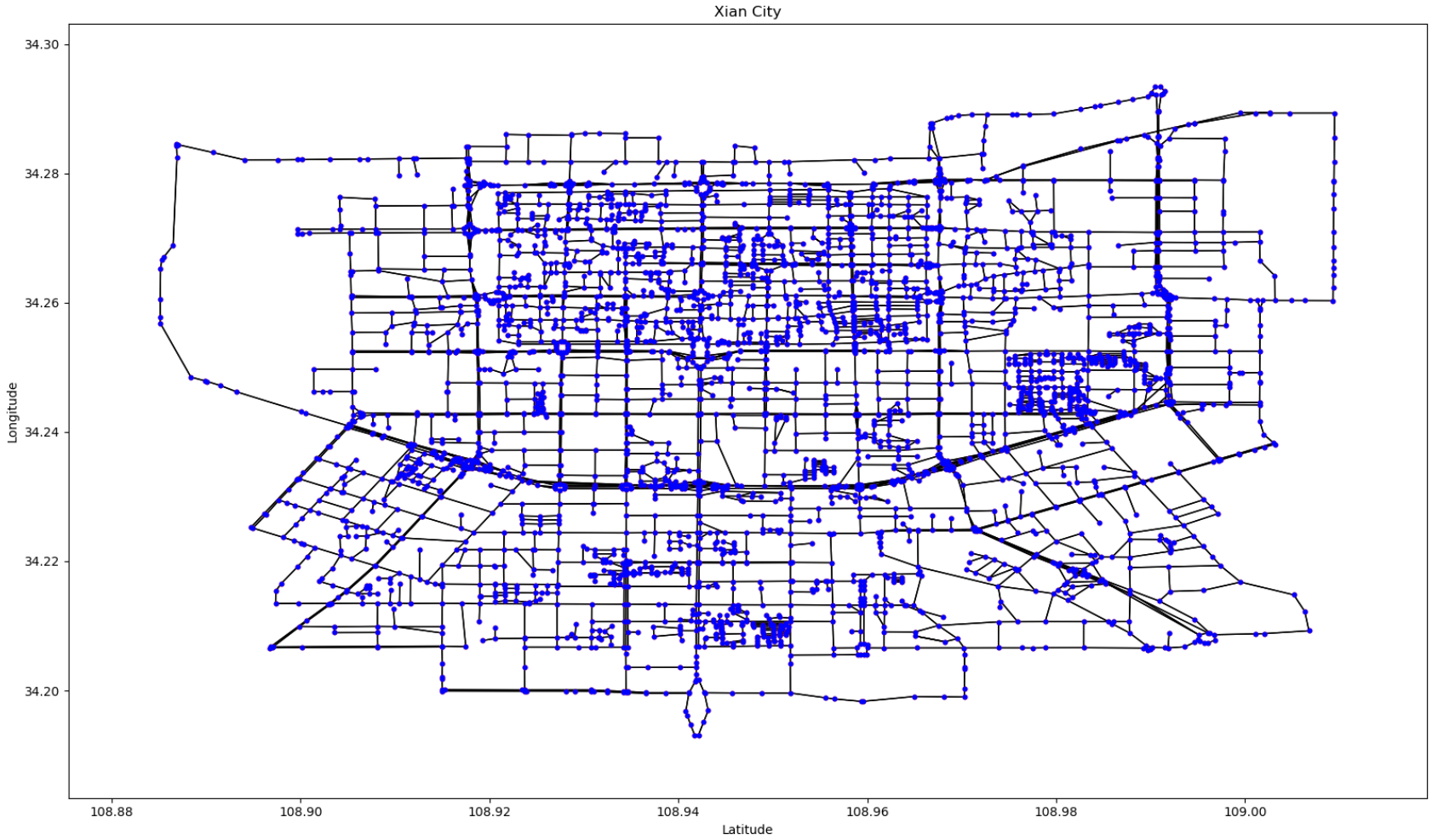
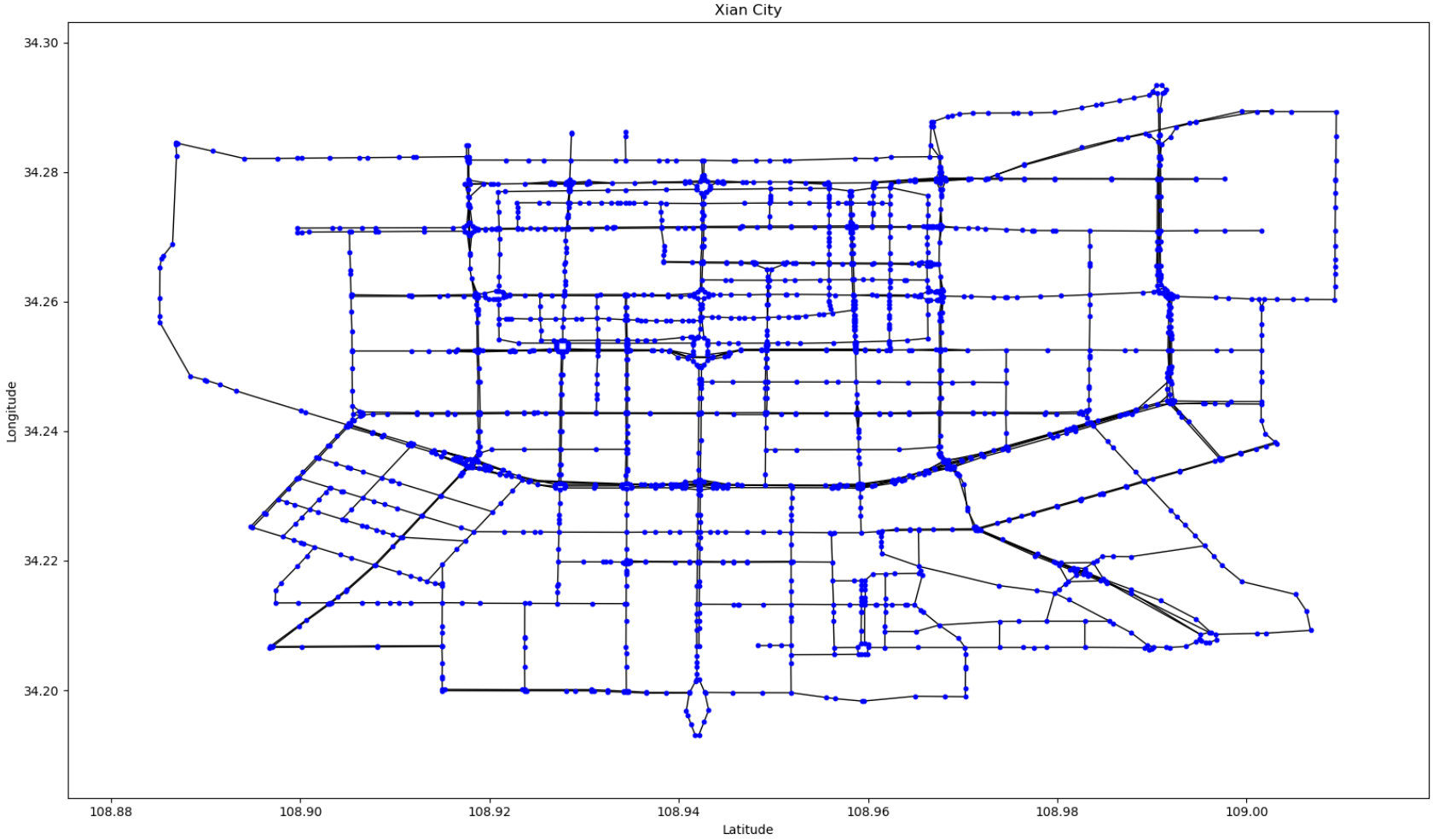


Figure 8: Zoomed simplification, A: All road segments B: primary, secondary and tertiary C: primary, secondary



A)

Universiteit Antwerpen Loughborough University Deusto Universidad de Deusto University of Deusto université de BORDEAUX



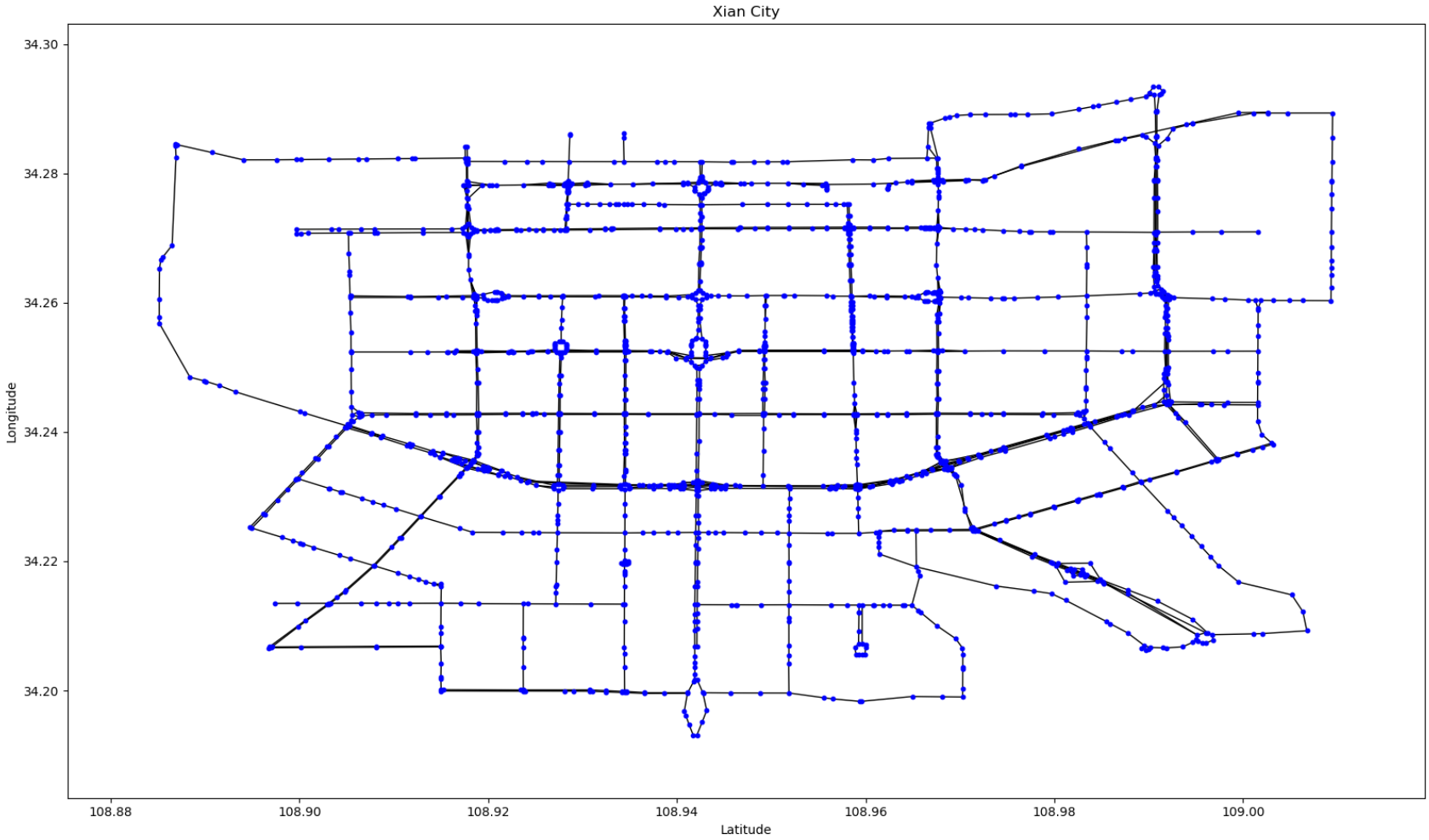
B)

Universiteit Antwerpen

Loughborough University

Deusto
Universidad de Deusto
University of Deusto

université de BORDEAUX



C)

Figure 9: Xi'an road network A: all road segments, B: primary, secondary and tertiary, C: primary and secondary

3.1.5.2 Graph simplification

Following the hierarchal road selecting a simplification can be performed. During the map-matching, edges are defined between two nodes. Every initial node has at least three connecting edges. The hierarchal road selection strategy will remove certain edges, and some nodes will have only one input and output. These nodes can be simplified to reduce the size of the graph. The process of simplification is shown in Figure 10. Formula (1) and (2) express the simplification. In these equations, v_i represent speed in a segment while n_i represents the number of measured cars in that segment. This operation is done for every timestep using matrix multiplications. This simplification can greatly reduce the size of the graph if combined with primary and secondary hierarchal road selection. These single input/output nodes exist due to the hierarchal road selection and thus this filter can only be used with hierarchal road selections. For instance, Xi'an starts with 2121 roads after hierarchal selection and reduces to 1320 if both filters are applied. For Chengdu, these values are 2343 and 1539. The downfall of this approach is that the real road structure is changed with virtual roads. This virtual data does not represent the real traffic situation and thus is less reliable. Due to this downfall, this filter is only briefly examined and will not be considered in this thesis. In my opinion, this approach has some potential where computational resources are sparse.

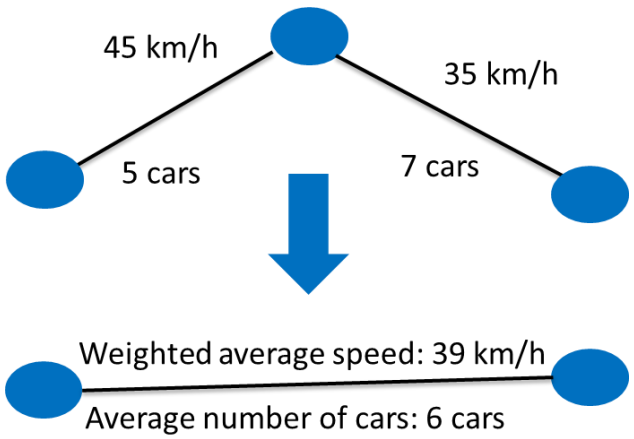


Figure 10: Example of graph simplification

$$v_{result} = \frac{v_{edge1} * n_{edge1} + v_{edge2} * n_{edge2}}{n_{edge1} + n_{edge2}} \quad (1)$$

$$n_{result} = \frac{n_{edge1} + n_{edge2}}{2} \quad (2)$$

3.1.5.3 Correlation and time-based correlation

Correlation-based approaches can be used to reduce the size of the data further. Looking for the association between segments can greatly improve predictions. A general overview of the correlation between two variables is shown in Figure 11.

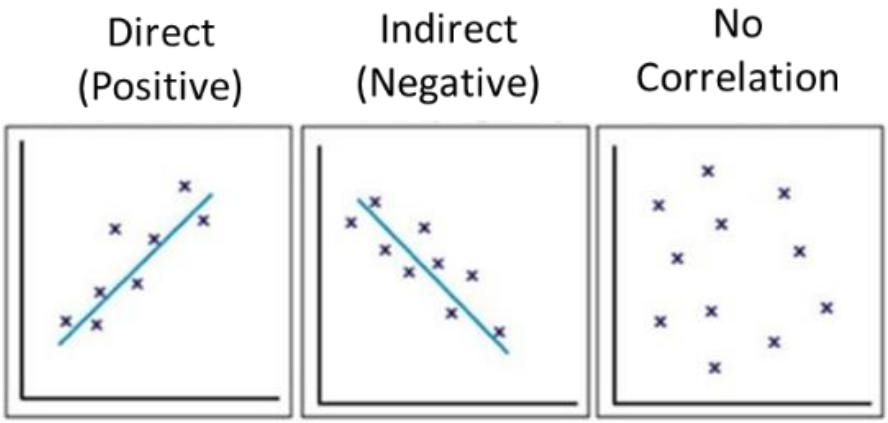


Figure 11: General overview correlation

Instead of using all data, which would increase the computational power and complexity, correlation-based filtering gives valuable inputs with a predefined size. Finding relations between road-segments is of great importance for predicting unseen irregular traffic patterns. Two correlation-based approaches will be used in this thesis. The first approach is to correlate the train set of the time series from the segment to predict against all other segments in the network.

With this general correlation, a selection of segments to include can be made based on the sum of the correlation of both features. The number of segments to include can vary based on the application. In these experiments, the number of segments is fixed to the top 100 most correlated. The second approach is like a regular correlation with the difference of including time. A regular correlation is made based on a shifted series of the segment of interest up-to 48 timesteps in the future, compared to all other segments, which are shifted 24 timesteps in the past. From this general correlation, a more sophisticated selection can be made.

For multistep long-term predictions, the selection of segments will be distributed based on individual correlation for each time step and the number of past timesteps that are included in the model. The final step of this process is shown in Figure 12. The number of selected segments per timestep is calculated with the formula (3). The process is broken if all timesteps to predict for are covered. If there is a remainder from the division, the remaining overall most correlated segments are included to guarantee 100 segments are included. The outcome of both approaches is shown in Figure 13; in this figure, part A represents the result of the normal correlation and part B represents the result of the time-based correlation. In both cases, the segment of interest (to predict for) is marked in green and the extracted edges after the procedure are marked in red.

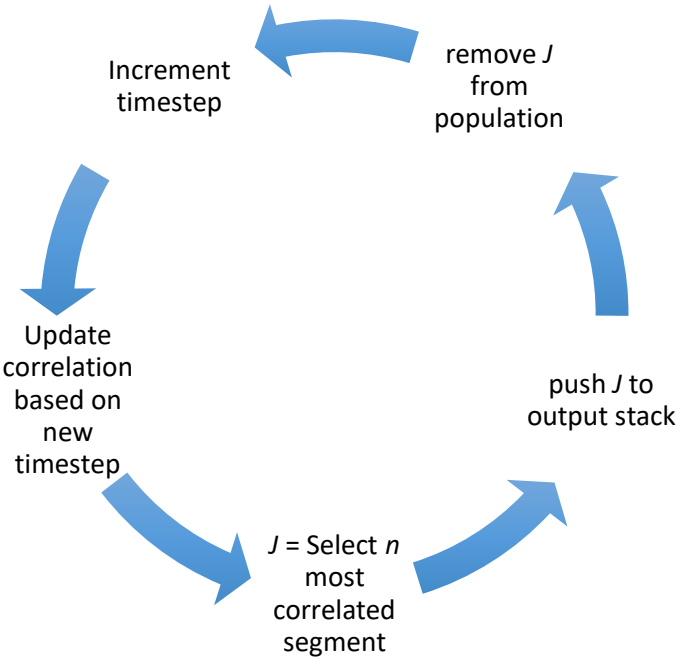
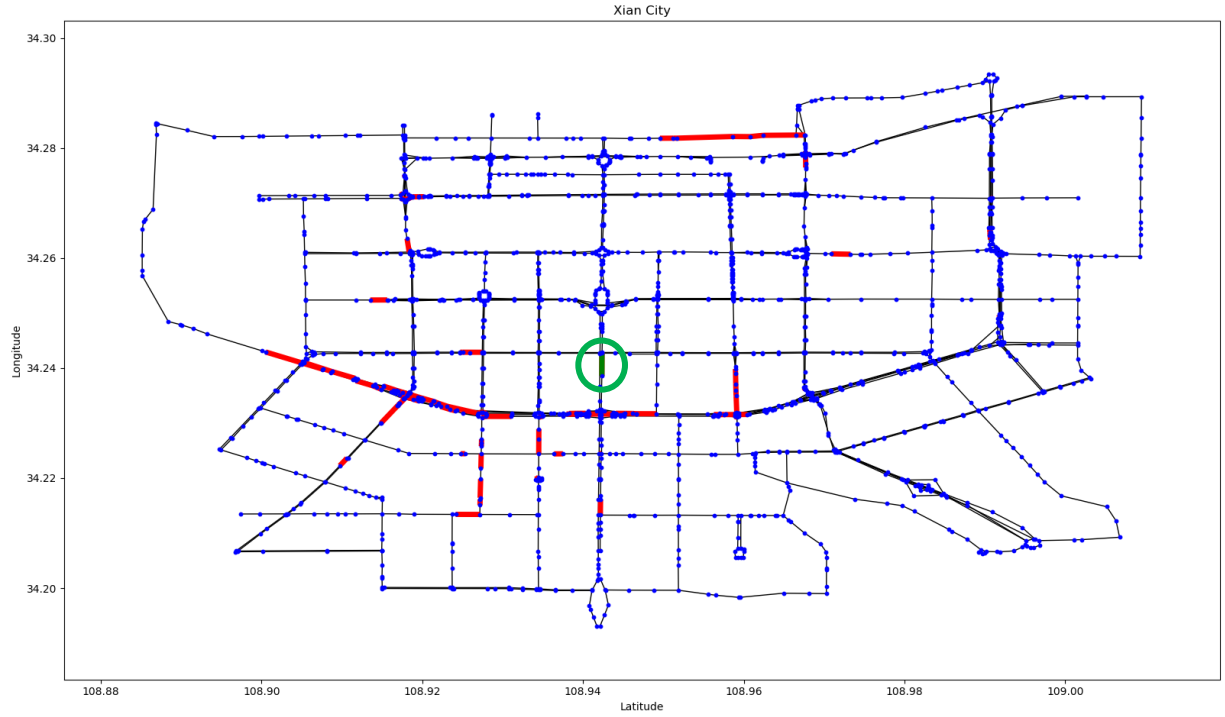


Figure 12: segment selection based on long-term multistep predictions

$$n = floor\left(\frac{total\ number\ of\ segments\ to\ include}{number\ of\ timestep\ to\ predict}\right) \quad (3)$$

A)



B)

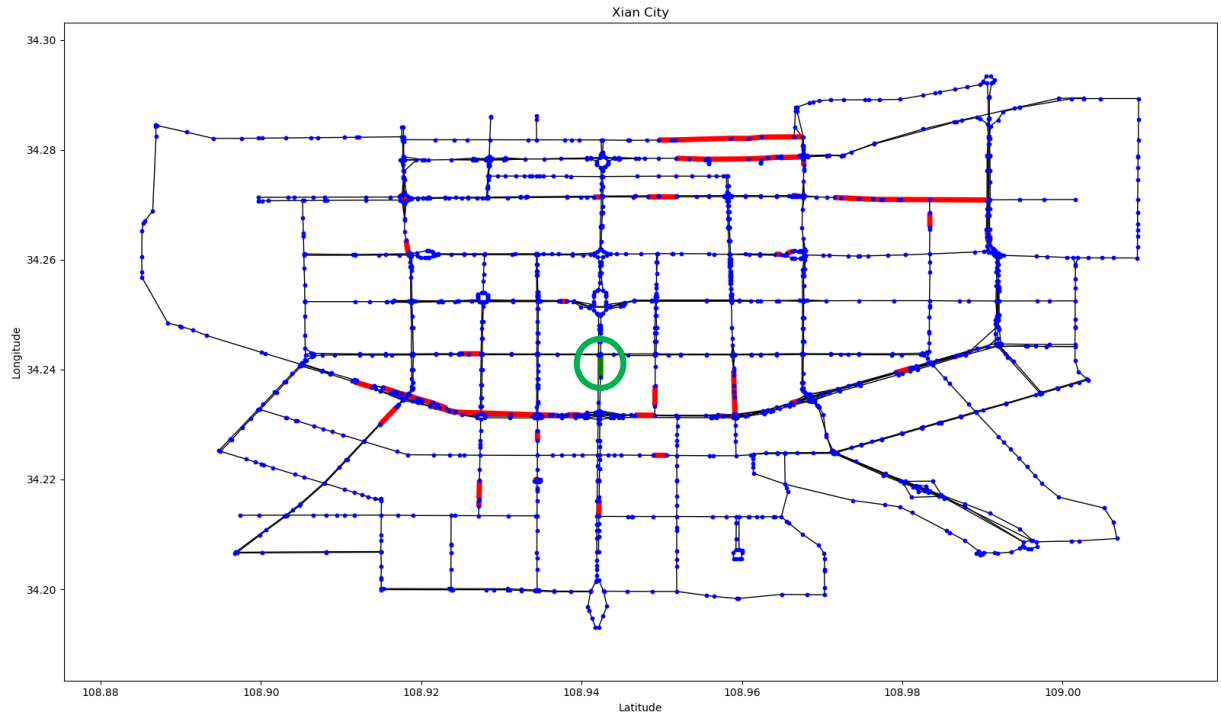


Figure 13: Correlation-based selection, red: selected, green: to predict for; A) standard correlation; B) time-based correlation

3.1.5.4 *Traffic flow in-out*

The last approach to reduce the size of the data is the use of traffic flow theory. This strategy includes the inflowing and outgoing roads to make predictions. To compare strategies, the number of segments to include is fixed on 100 as with correlation-based approaches. A visualisation of this approach is shown in Figure 14. The selection is made using tree-like structures growing from the segment of interest. Within this process, duplicates are removed until 100 segments are selected.

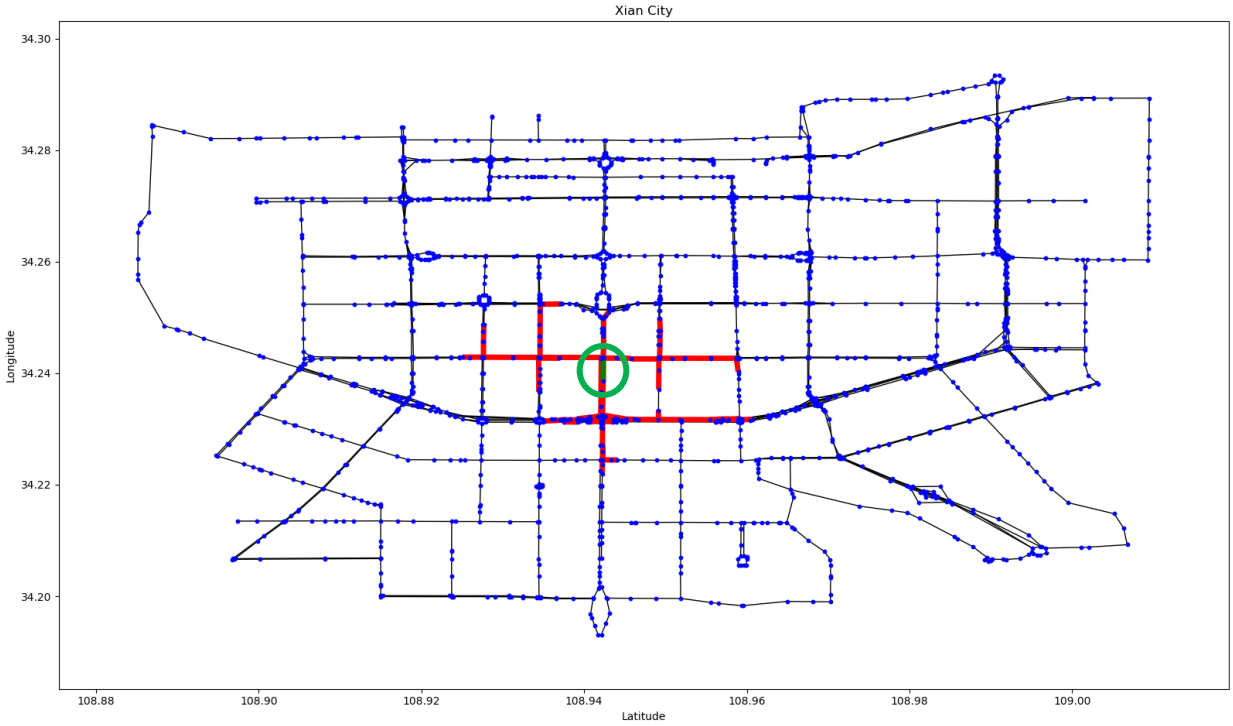


Figure 14: Traffic flow in-out, red: selected, green: to predict for

Due to map-matching, some areas have more nodes or are denser compared to others. These smaller edges are created during map-matching. As shown above, the outgoing tree structure covers more surface compared to the ingoing. On the south side of the segment is a busy intersection connecting the road to a highway resulting in more nodes and edges. These nodes and edges contain mainly similar information and thus are a downfall of this approach. Including edges with the same information would decrease the quality/size ratio of the data structures.

3.1.6 Adjacency matrix creation

From the reduced graph structure, an adjacency matrix is extracted. The matrix extracted from $N \subset L$ is square with dimensions $|N| \times |N|$ with L representing all nodes and N the nodes connected to the segments to include. For each feature in the graph, a different adjacency matrix is extracted. These matrices have identical shapes and are stacked to form a three-dimensional matrix with features on the third axis and thus can be interpreted as an image. An image has values on all pixels, while the information in the adjacency matrix is rather sparse. To keep the spatial relation, the matrix is sorted by latitude and longitude. This approach results in a sparse matrix with some clusters representing key road clusters in the city, as shown in Figure 15. The adjacency matrix unsorted is not identical over the first diagonal because this work uses a bidirectional graph network to represent the traffic situations. If two nodes contain multiple streets between them, both streets will be included on each side of the diagonal. With this method, there are actually more than 100 segments included in the matrix without increasing its dimensions.

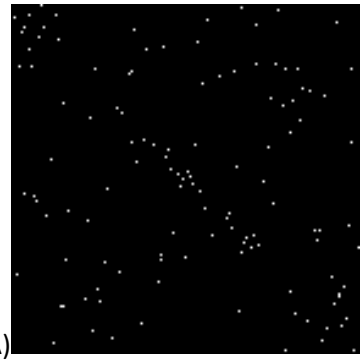
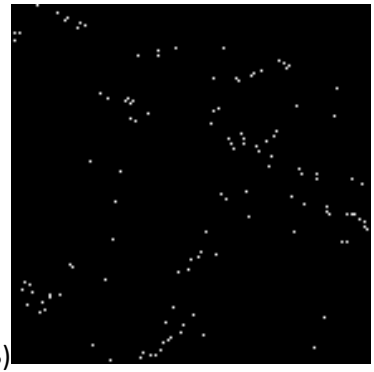
A)  B) 

Figure 15: Adjacency matrix; A) unsorted; B) sorted

3.2 Network architecture

This section presents the proposal of deep learning architecture used in this work. Given the variety of elements used, this section is structured as follows: first by a general description of the hybrid model will be discussed in section 3.2.1, followed by individual clarification of each aspect.

3.2.1 General description

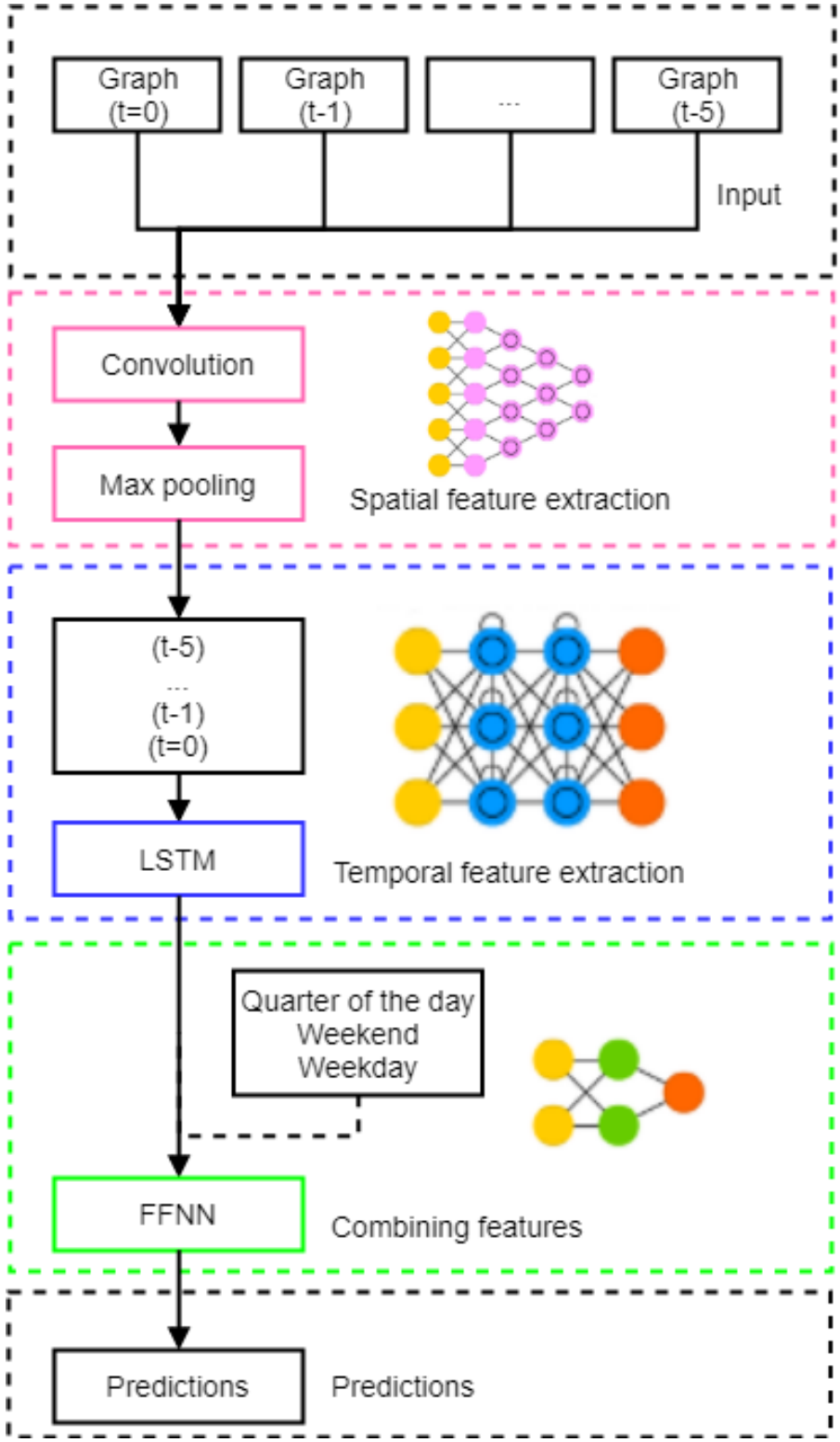
This work proposed a hybrid Deep Neural Network (DNN) architecture, that aims at finding both spatial and temporal relations that appear in traffic patterns. Concretely, it is composed of three main components:

(i) Graph-convolutional neural network

(ii) Recurrent neural network

(iii) Feedforward neural network

Figure 16 illustrates the model's general structure. The model uses historical information considering the *n* last time steps of data provided in adjacency matrices. These structures are generated in order to consider spatial dependencies of the values of traffic, which can be interpreted by the model. Once the spatial information is processed, the recurrent unit is fed with the extracted tensor. This stage evaluates the previous *n* time steps of the spatial relations with the aim of extract the possible temporal trends. Finally, results are gathered into a Feed Forward Neural Network (FFNN) and concatenated with possible contextual information to produce the final output. Each of the blocks that compose the architecture is described in following subsections.



585

Figure 16: General structure of the proposed model

3.2.2 Introduction to Convolution Neural Networks

Figure 17 illustrates an example of a convolution layer. Convolution layers are the base of a convolutional network. These networks are designed to work in computer vision. If fully connected networks are used, the number of parameters is in the billions for processing common images. Convolution is cropping an image to match a given size and applying an element-wise matrix multiplication between the cropped image and a filter. This multiplication is followed by a summation to form the final float variable. The cropping of the image depends on the size of the given filter. This filter is first applied to the cut-out matrix at the top left corner. After the first calculation, the filter slides along an axis with a given stride. This process continues until the entire matrix is covered by the filter. The trainable parameters in these networks are the weights of the filter.

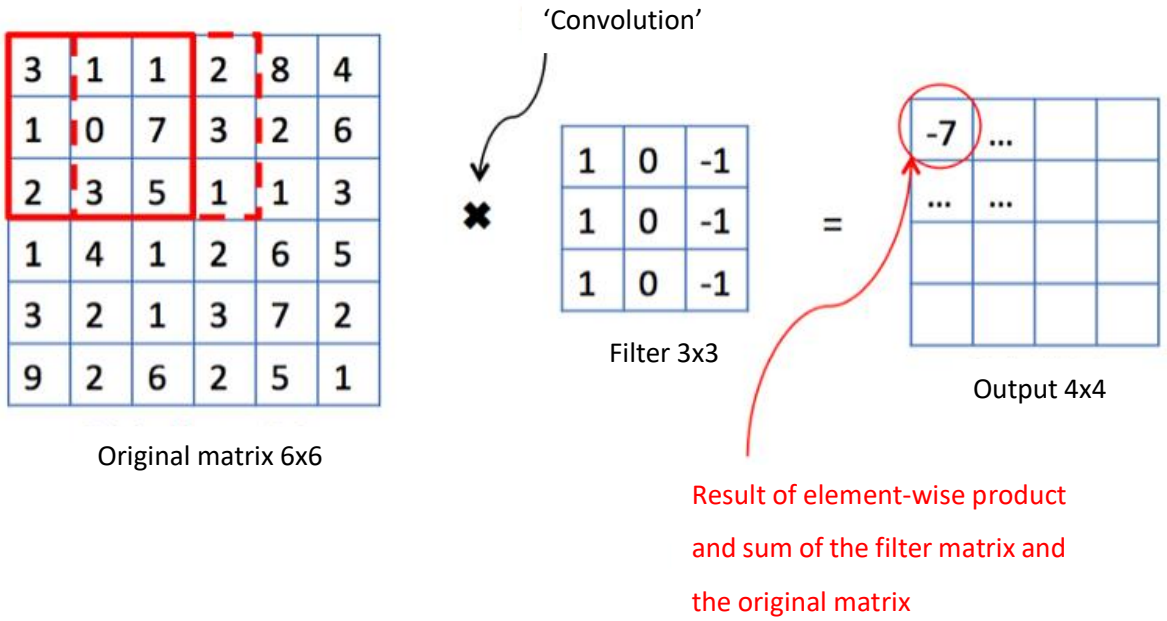


Figure 17: Introduction to convolution, source[3]

[3] https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

Max pooling is usually combined with convolution layers to reduce the size of the image and maintain some information. Like convolution, max pooling uses a filter that slides over the matrices. Max pooling, like its name suggests, only keeps the maximum value from the filter. Figure 18 illustrates this process; the filter has 2x2 dimensions and a stride of 2.

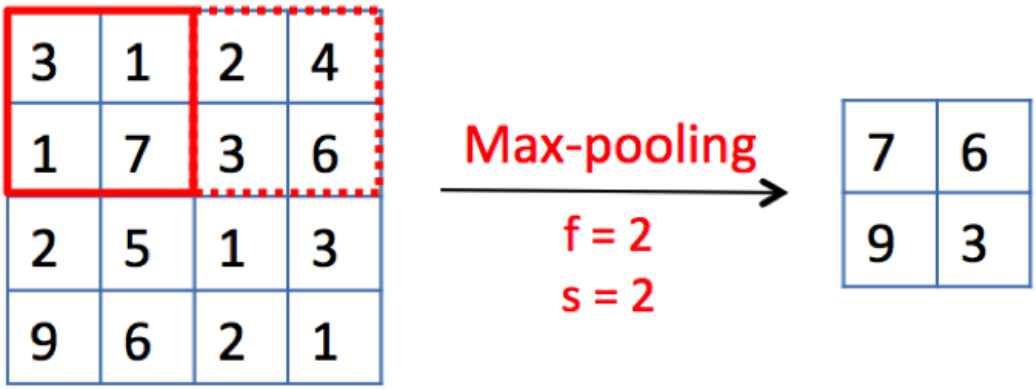


Figure 18: Introduction max pooling, source4

3.2.3 Graph Convolution Neural Network

Extracting spatial dependencies from traffic data is an important challenge in TF. Traditional CNN's can detect spatial relations in data types such as images or other two-dimensional stacked data types. The complexity of a city's road network makes it difficult or not possible to represent the information in a two-dimensional matrix. These complex road networks can be represented with the use of graph structures, considering streets as edges and intersections as nodes. Figure 9C shows a graph structure of Xi'an with only primary and secondary roads. There are two general approaches to perform convolution on these graph structures to form a structured data format. Firstly, [37] introduces a spectral environment to the graphs using the Fourier transform. Second, [38] expands the spatial definition of the graph structure. This work will use the latter. This transformation consists of constructing an adjacency matrix. For a graph with node set L, the adjacency matrix A has a shape of $|L|x|L|$. For every A_{ij} there is either a value of the weight of the edge from node i to node j, meaning A_{ij}=0 that there is no connection between the nodes. This matrix can be interpreted by a CNN, extra features can be created with adjacency matrices and stacked to extend the feature space. Using this approach, spatial features of complex road networks can be codified in a way that can be processed by CNN's.

[4] https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

In this work, two different convolution methods will be used. Firstly, a repetition of convolution layers followed by max-pooling layers (CMCM). The number of filters is increased in the second repetition. This method is used in applications with images and general convolution applications. With this strategy, first microfeatures are extracted followed by macro feature extraction in the second repetition. Secondly, as adjacency matrices are rather sparse in this application, this approach first reduces the size of the adjacency graph by 'zooming in' on the clusters using max pooling (MMCCM). This process is shown in Figure 19, the overall average speed at night is higher, resulting in a whiter adjacency matrix. The size of the passing tensor is greatly reduced and thus lowering the number of parameters resulting in faster training and reducing complexity, as shown in Table 3. These models are called shared vision models because they have multiple instances sharing weights. In this case, the models are used on all *n* historical inputs with shared weights, meaning that the model will focus on the same features on each time step. The output tensor size is significantly smaller in MMCCM; this output is then fed into the recurrent net. This connection is where the number of parameters differs, resulting in faster training. CMCM has around 19 million parameters in this connection, while MMCCM 2 has 1.5 million.

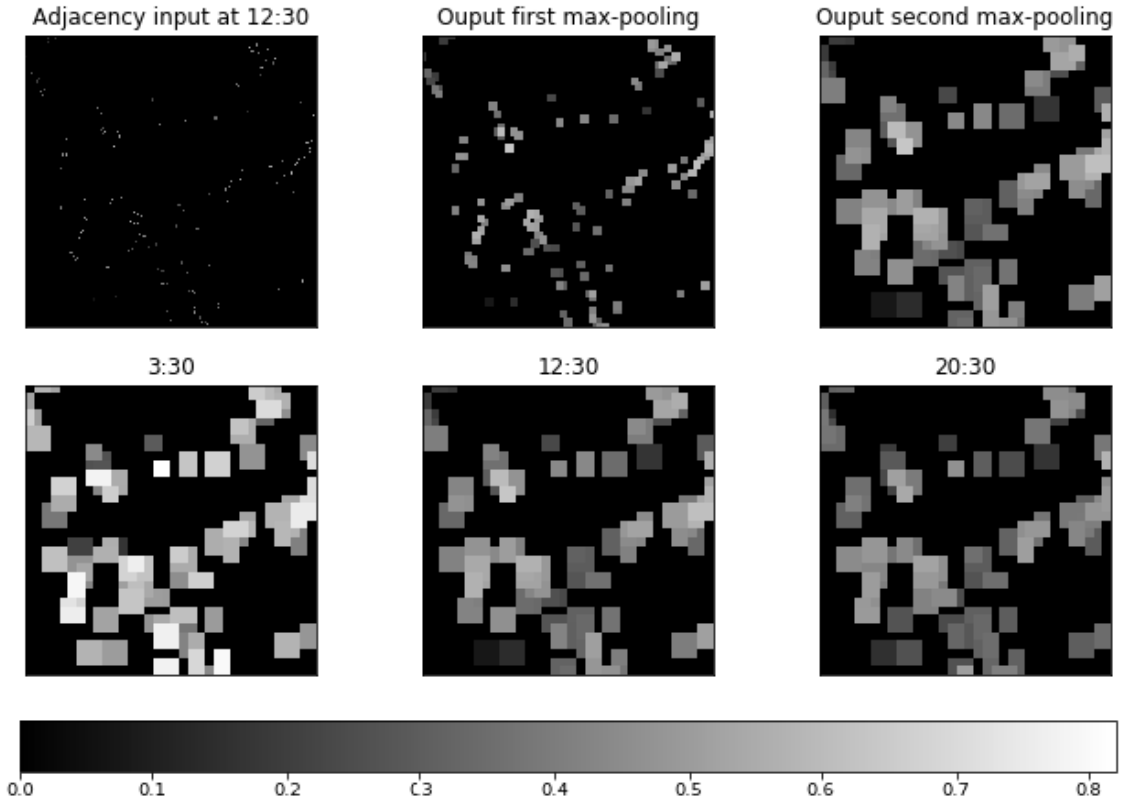


Figure 19: Adjacency matrix inputs

CMCM	main parameters	Pool	strides	activation	size
Convolution2D	filters=5	(3,3)	(1,1)	Relu	(5,132,132)
Maxpooling	filters=15	(4,4)	(2,2)		(5,65,65)
Convolution2D		(3,3)	(1,1)	Relu	(15,63,63)
Maxpooling		(2,2)	(2,2)		(15,31,31)
Flatten					(14415)
Dropout	0,2				**(14415)**
MMCCM					
Maxpooling		(4,4)	(2,2)		(5,66,66)
Maxpooling		(4,4)	(2,2)		(5,32,32)
Convolution2D	filters=5	(3,3)	(1,1)	Relu	(5,30,30)
Convolution2D	filters=5	(3,3)	(1,1)	Relu	(5,28,28)
Maxpooling		(2,2)	(1,1)		(5,14,14)
Flatten					(980)
Dropout	0,2				**(980)**

Table 3: Shared vision models

3.2.4 Introduction to Recurrent Neural Networks

Recurrent neural networks try to make use of sequential information. In traditional neural networks all inputs are independent, for sequential data traditional networks do not capture all the information. If you want to make a prediction based on a sequence of information, knowing the order of the previous samples will improve the predictions. Recurrent neural networks are called recurrent because they perform the same operation for every element of a sequence, with reusing the output of the previous computation as an input in the following node. This process is illustrated in Figure 20. The figure unrolled the recurrent network to an unfolded version. This unfolded representation represents how the computations are linked in the sequence. The parameters shown are described as followed:

- x_t is the input at time step t.
- s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1})$.
- o_t is the output at step t.

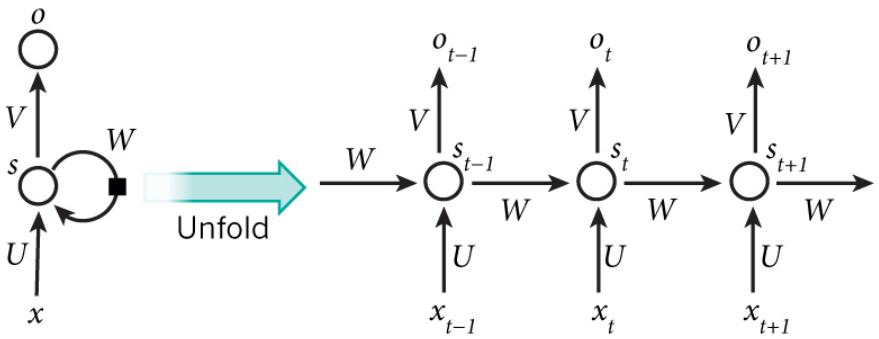


Figure 20: Introduction to recurrent neural networks, source[5]

The hidden state captures information from previous steps in the sequence. This hidden state works as a sort of 'memory'. The weights learned by the parameters (U, V, W) are shared for each instance of the recurrent network, which reduces the total number of trainable parameters. Based on this general implementation, some more advanced models are created. The LSTM, which is used in this thesis, is one of the more advanced cells. Figure 21 gives a good overview of the working of these cells. The cell might be complex to analyse so an abstraction will be used to explain its working.

If examining the figure, there are three inputs and two outputs. The cell uses the input of the current timestep, the memory of the previous unit and the output of the previous unit to do an internal update and provide some output. This internal update will update its memory; this memory is then combined with the current input and memory of the previous cell to another decision maker, creating the second output. The cells can pass information of previous timesteps with this pipeline of memory passed. On its journey, the information can get some addition or subtraction of information. These additions and subtractions are trainable, so the model can learn what kind of information is relevant during training.

[5] http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

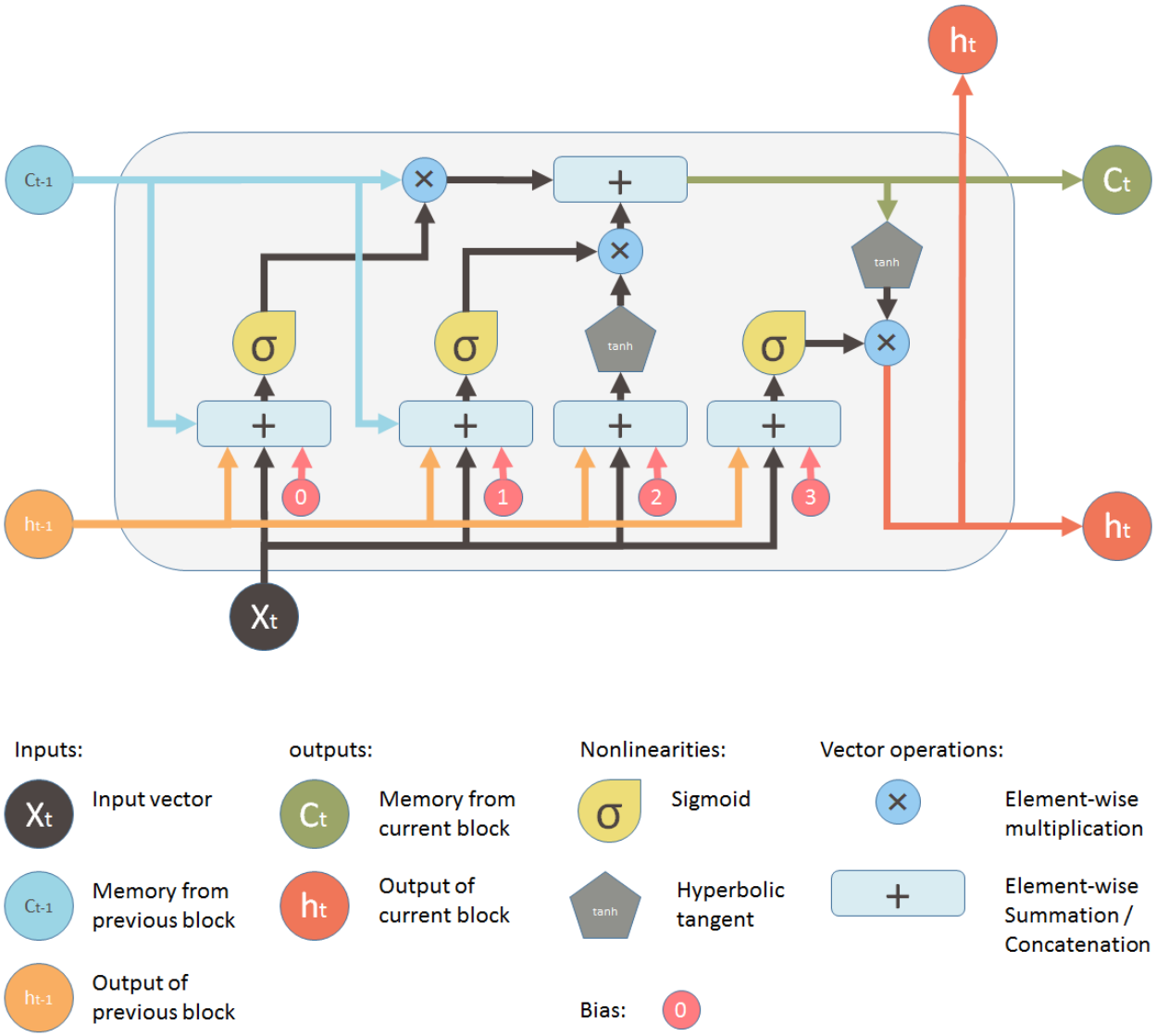


Figure 21: Introduction LSTM, source[6]

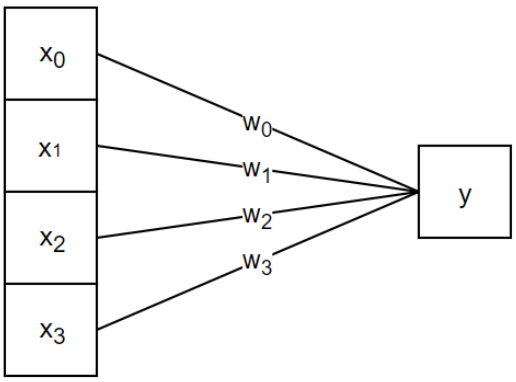
[6] https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

3.2.5 Long-Short term Memory Neural Network

Like spatial relations, temporal relations are key features to consider in TF. This relation can be extracted from sequential data with the use of RNNs. These networks are highly complex and tend to show some flaws in traditional implementations [39]. Because of these flaws, an LSTM network is used. This network has more resistance to exploding-disappearing gradients but is not immune. LSTM networks can memorise historical data for longer periods while also consider the current values. These networks have proven good performance in the field of text processing and time series analysis, among others. Traffic data can be mostly repetitive since most people have a similar commute during the week and different behaviour during weekends. Because of this repetitiveness, the use of these type of networks makes sense as they can memorise certain events with their consequences. An example of a repetitive event is the daily rush-hours; these hours are of high interest as they have the largest negative deviation concerning the general traffic situation. During these events, the traffic state is generally at its worst. Being able to Predict these events is the first step in preventing them.

3.2.6 Introduction to Feed Forward Neural Networks

Deep learning is a collective name for a family of neural networks. The neural networks differ in complexity and architecture. The most common and relatively easy network will be explained. These networks are called MultiLayer Perceptron's (MLP) or also called feed-forward neural networks. MLP's can be considered as a combination of multiple linear models. Linear models use the weighted sum to predict a value. An example of a linear model and the related formula is shown in Figure 22. In this figure, *x* denotes input features while *y* represents the output, which is calculated by the weighted sum of the inputs added with a bias *b*.



$$y = w_0 * x_0 + w_1 * x_1 + w_2 * x2 + w_3 * x_3 + b$$

Figure 22: Linear model example

MLP's repeat this process multiple times in layers which are called 'hidden layers'. An example of a basic MLP is shown in Figure 23. Combining weighted sums results in a weighted sum, to make this approach more valuable a non-linear function can be applied to the result of weighted sums. With this non-linear function, the models can learn more complex relations which are not possible with standard linear models. MLP's have trainable weights and biases, in regression applications, the main objective is to reduce the error of a loss function. Deep learning uses optimisers, which will change the parameters depending on their influence on the loss function to optimise the error between predictions and real values. This optimisation is performed on a training set and validated on a testing set. Splitting training and testing ensures the model to generalise and prevent overfitting.

Some more keywords in deep learning are:

- Sample: One element of the data.
- Batch size: a set of samples to evaluate before an optimisation step is performed.
- Epoch: One pass over the entire training set
- Activation function: non-linear function applied to weighted sums.
- Supervised learning: A problem with known inputs and outputs. The output is predefined and is known during training.

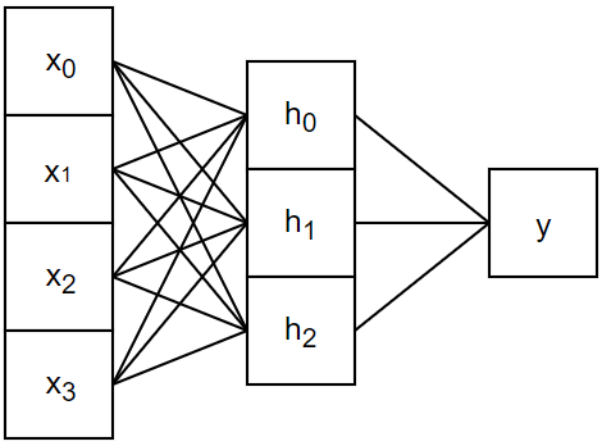


Figure 23: Example MLP with one hidden layer

3.2.7 Feed Forward Neural Network

These types of networks are the most common in DL. Each neuron in a layer is connected to all neurons of the following layer. Stacking these layers reveals higher dimensional relations within the input data, as explained in detail in section 3.2.6. Regarding forecasting, the use of these networks can be use-full in evaluating the extracted features with some extra information. This extra information can contain a variety of features such a measure of time to give the model a sense of time. As mentioned before, the repetitive nature of traffic data might be linked with a measure of time to increase the accuracy of long-term predictions.

3.2.8 Temporal Graph Convolutional Network

The make full use of both temporal and spatial relations, the extracted spatial features are used as inputs to the LSTM network. A predefined number of sequential historic adjacency matrices is first fed to the CNN to extract the spatial features. These features are stacked by time and fed to the LSTM evaluating the time-related trends in the spatial features. In this manner, it is possible to include both temporal and spatial relations at the same time. The combined extracted features are used as the input of the FFNN. In this part of the model, the prediction is formed. To guide the model in the right direction, extra features can be concatenated before the FFNN input. For long-term multistep predictions, the output vector will be equal to the timesteps to predict. Each output will represent a given timestep.

3.3 Experimental set-up

This section will elaborate on the evaluation process in section 3.3.1 and section 3.3.2. This is followed by a more detailed description of the hyperparameters of the proposed models in section 3.3.3.

3.3.1 Evaluation parameters

To evaluate the performance of the proposed pre-processing, a comparison will be conducted using the suggested models with different methods. The performance of the models will be compared to traditional algorithms such as k-nearest neighbours (K-NN), regular LSTM and Support Vector Machine (SVM). For more advanced algorithms, I refer to those presented at the TRANSFOR19 competition described in Section 4.4. All methods are evaluated in terms of four measures: Root-Mean-Square-Error (RMSE), Mean-Square Error (MSE), Mean-Absolute Error (MAE) and Mean-Absolute-Percentage Error (MAPE), whose formulations are presented in Equations (4), (5), (6) and (7). In these formulas, r_i and p_i represent the real and predicted value respectively, w_i represents the weight of the predicted value and n number of samples. These performance indicators will have three versions:

- Regular implementation: where all the samples in the data have the same weights $w_i = 1$
- Weighted implementation for rush-hours: where samples registered outside rush hours (6.00-11.00 and 16.00-21.00) have $w_i = 0.5$ and samples within the rush hour $w_i = 1$.
- Windowed implementation for rush-hours: samples within the rush-hours have $w_i = 1$ while samples outside rush-hours are not considered in the calculation and thus have $w_i = 0$

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} w_i (r_i - p_i)^2} \quad (4)$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} w_i (r_i - p_i)^2 \quad (5)$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} w_i |(r_i - p_i)| \quad (6)$$

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \frac{w_i |(r_i - p_i)|}{p_i} \quad (7)$$

All methods are optimised using Root Mean Square Propagation with loss function being weighted MSE. This optimiser is most commonly used in regression applications. The models are trained for 50 epochs with a call-back to the epoch with the best score on the test set. The batch-size is variable depending on the machine executing the program to maximise efficiency. Figure 24 shows the number of observed cars aggregated for one week; the amount of measurements during night-time is sparse, resulting in untrustworthy data. The weighted evaluation will force the model to focus on the time intervals of interest. This strategy helps the model focus on data that is more reliable and lowers the influence of the missing values during night-time.

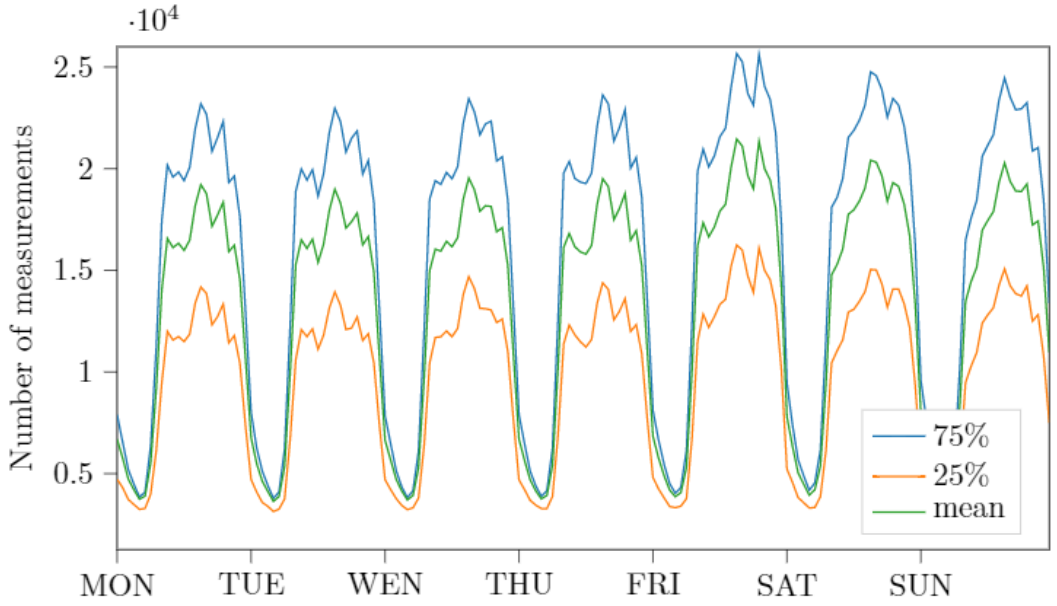


Figure 24: Number of measurements aggregated in Xi'an

3.3.2 Validation segments

As validation, multiple road segments are used during the evaluation from both Xi'an and Chengdu; these segments are chosen based on their standard deviation and speed profile to cover different traffic sequences. Other criteria the segments should fulfil are primary or secondary road, a distance over 100m, less than 20% missing values and more than 130k measurements. Table 4 shows the properties of the selected segments. Five segments are selected from each city with two extra segments in Xi'an, these two extra segments (2748, 160) were the roads to predict for in the TRANSFOR19 forecasting competition. The results of these two segments can be compared to the results of other approaches used in this competition; segment 2748 is commonly named as 'north segment' and 160 as 'south segment'. Figure 25 and Figure 26 highlights the selection of both cities.

Chengdu	Mean [km/h]	std	Win_mean [km/h]	Win_std	Samples	Missing [%]
207941	25.28	3.53	24.43	3.02	189677	5
7171	28.38	4.66	27.42	4.31	130234	9
9273	54.25	11.69	49.77	13.96	187936	6
210521	23.39	9.11	20.64	9.72	147525	9
128	26.91	5.95	25.97	5.68	130881	12
Xian	**Mean [km/h]**	**std**	**Win_mean [km/h]**	**Win_std**	**Samples**	**Missing [%]**
161	26.16	8.87	22.62	6.50	139258	17
2747	23.67	4.93	22.89	4.33	190621	16
92053	22.07	3.73	20.79	3.21	203041	16
7532	32.23	11.97	26.97	10.32	237328	16
7524	36.74	8.70	33.43	9.74	159451	20
160*	16.92	6.50	15.15	5.44	137235	17
2748*	25.43	9.80	21.46	7.76	170771	17

Table 4: Segment selection, win: windowed implementation. (*) denotes segments used in the TRANSFOR19 forecasting competition

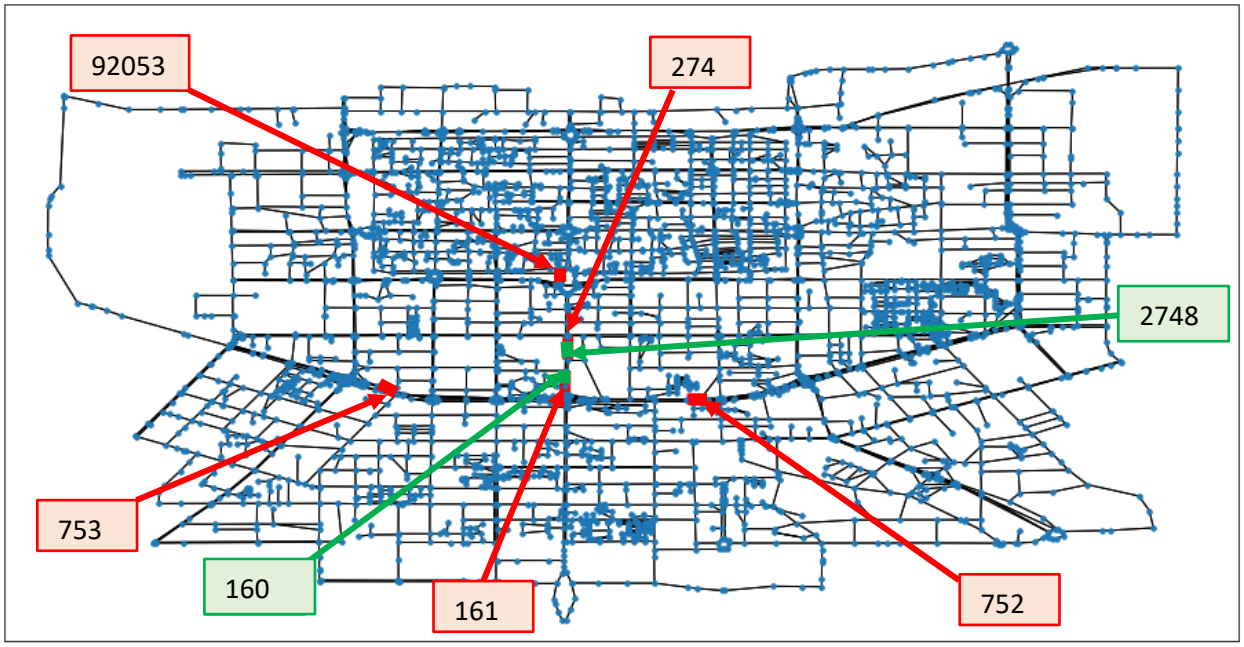


Figure 25: Segment selection Xian City, red: validation segments, green: competition segments

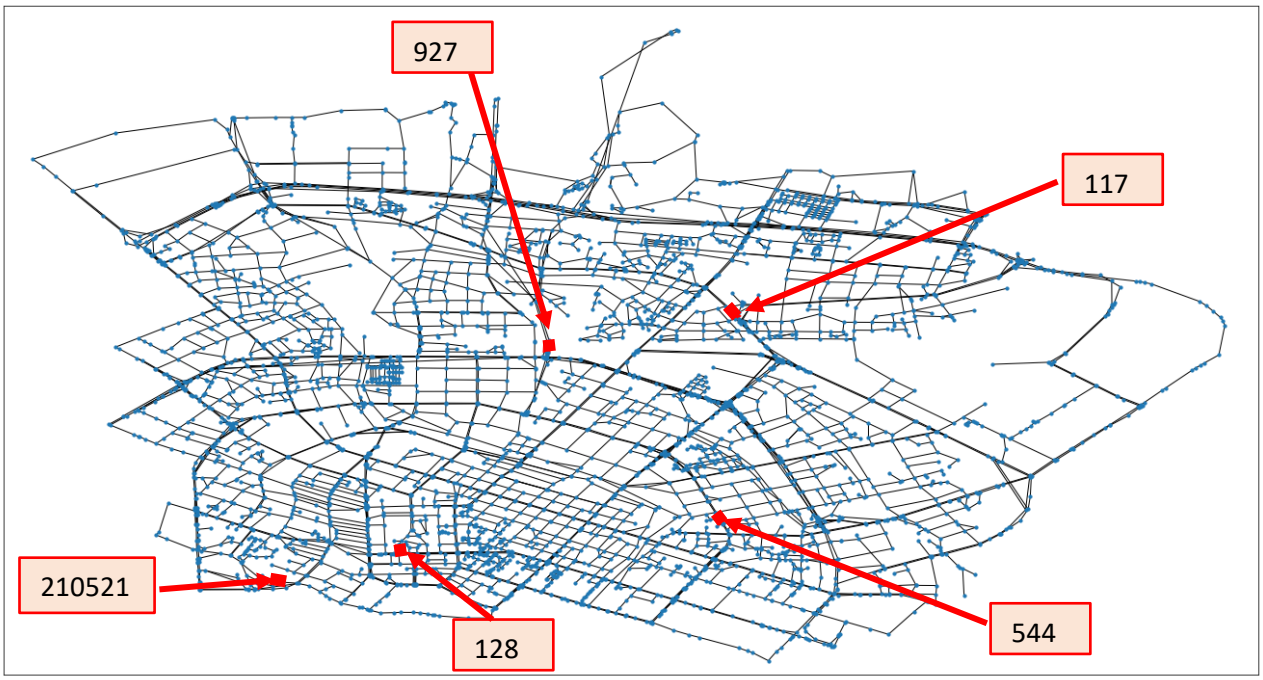


Figure 26: Segment selection Chengdu, red: validation segments

3.3.3 Network hyperparameters

Table 5 shows the hyperparameters for segment 128 from Chengdu. The size of tensors can slightly differ between segments depending on the shape of the input adjacency matrices. Five variants of the main architecture will be evaluated. First, regular implementation of convolution shown in Table 3 as CMCM. This convolution is followed by the LSTM's. At this point three variants are tested: (i) no extra input named Convmax, (ii) Convmax-Sigmoid is identical to the first model but has a sigmoid activation function on the output layer, (iii) extra inputs (weekend Boolean, past day average speed at the same time and quarter of the day) called Extra-input, (iv) Extra-input-rnn uses aforementioned extra inputs adding a small LSTM of five nodes that evaluates the past hour of the average speed related to the output segment. The fifth variant (v) uses Shared vision model MMCCM shown in Table 3 without extra input named Maxconv.

Name Layer	Main parameters	Activation	Dimension of Tensor
Input			(5,2,134,134)
Shared vision model	CMCM / MMCCM		5*(14415) / 5*(980)
Concatenate			(72075) / (4900)
Reshape			(5,14415) / (5,980)
LSTM	N=250, rdropout=0.2	Relu	(5,250)
LSTM	N=250, rdropout=0.2	Relu	(5,250)
LSTM	N=250, dropout=0.2	Relu	(250)
Concatenate	**Optional extra inputs**		**(254)**
Dense	N=150	Relu	(150)
Dropout	0.2		(150)
Dense	N=150	Relu	(150)
Dropout	0.2		(150)
Output	N=48	None / **Sigmoid**	(48)

Table 5: Hyperparameters models

All models take the last five timesteps in consideration, corresponding to 25 minutes of historical data, to make predictions for five minutes or four hours in multistep. This last prediction is a prediction for every five minutes in the next four hours. The number of parameters in the models differs between 4 million up to 20 million. All models using the MCMC model have high numbers of trainable parameters, as explained in section 3.2.2.

3.4 Sigmoid activation on the regression output

Convmax-Sigmoid described above uses a sigmoid activation function on the output. Almost all literature regarding regression application advises not to use activation functions on the output as this is usually only used in classification. After thoroughly analysing the data, I experimented using a sigmoid function as the activation on the output. In traffic data, the limited range to predict is known which is between zero and the allowed speed in that segment. Generally, it makes no sense to predict an average speed above the speed limit. In regular regression applications, no activation function is used. The sigmoid activation function does not allow the model to predict outside]0,1[. In other applications this would mean, depending on which scaler applied, that the model is not able to predict above or below the minimum or maximum of the training set. Knowing this limitation in TF, the sigmoid function gives the model more room to be precise compared to a linear (standard) approach. Figure 27 shows this difference, the histograms are created based on the value the model should provide before the activation to make the correct prediction. Most predictions for this segment are between 0.1 and 0.5 on the right y-axis. Within this area, the sigmoid has a lesser gradient than linear activation. This lesser gradient is what helps the model to be more precise expanding the range on the x-axis from [0.1,0.5] to [-2.2,0]. In general, the sigmoid activation function improves the symmetry of the output distribution and in turn, the performance of the method.

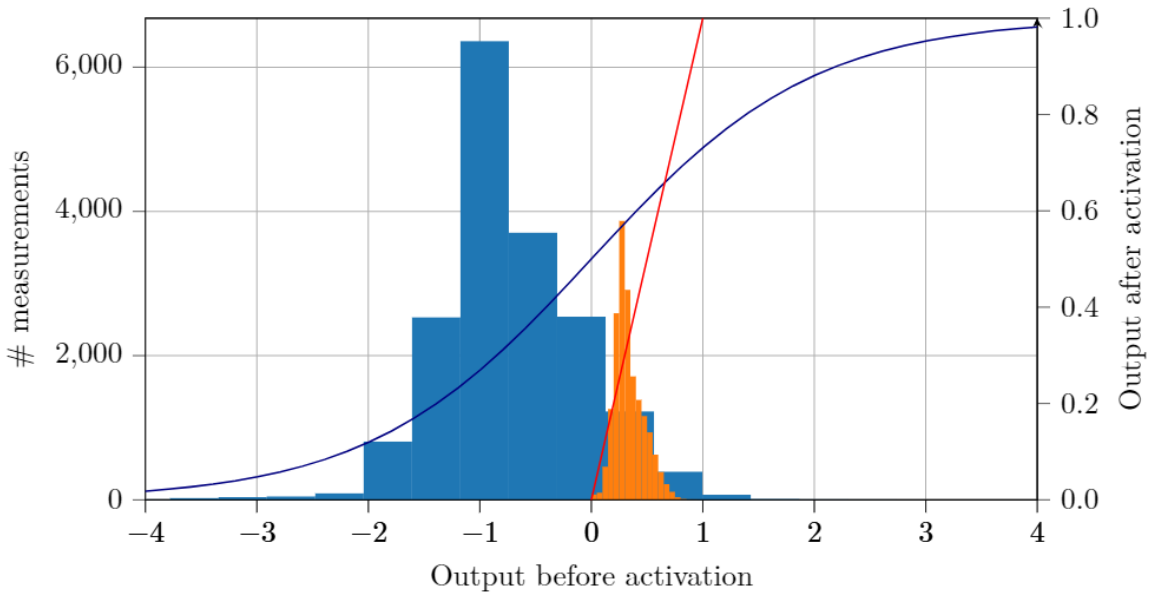


Figure 27: Sigmoid histogram; blue sigmoid implementation, red/orange no activation function

3.5 Networkwide long-term prediction model suggestion

This section will introduce my last advancements into a model which can make network-wide predictions over different time horizons. This model is based on the knowledge learned during the development of models mentioned above. The strategy of the model is to predict networkwide and reuse its predictions to predict following timesteps. The model is trained in a recurrent way to boost the performance of refeeding results. To simplify the process, the number of measured cars feature is removed, only the measured average speed will be considered. For this thesis, I have created some prototypes to prove the principle. Next segments will describe this three-stage-model.

3.5.1 Time-autoencoder

If predictions are made for one segment, one of the previous techniques can be used as they use the data of this segment and compare them to others. For networkwide predictions, the problem arises of which information to use as inputs. Using all information would be inefficient and only possible with enormous computational resources. To tackle this first obstacle, I looked towards autoencoders to 'encode' the state of the city's traffic network. This strategy seems promising, but autoencoders do not take time into account. As these models use *n* historic time steps, I made some changes to the architecture of an autoencoder to incorporate time. Figure 28 illustrates this adapted autoencoder.

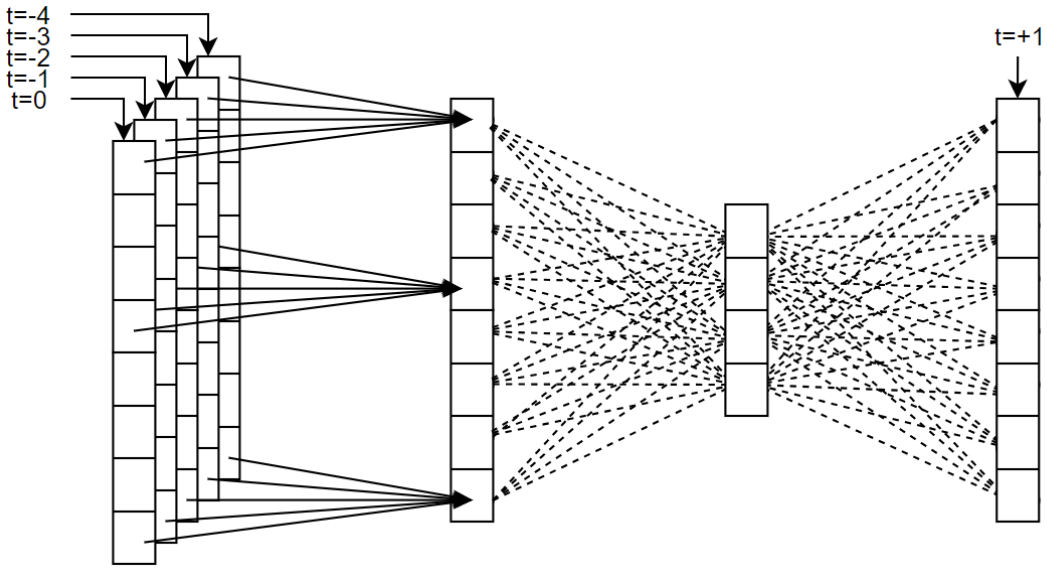


Figure 28: Time-autoencoder

The first layer of the neural network is a locally connected layer to reduce the entry of five timesteps to one value representing a combination of these five entries. The following two layers are fully connected layers with a reduction from the input shape to 200 nodes. The number of nodes can differ, based on the computational resources available to me, I decided to use this number. These 200 nodes are then connected to the output, which is the average speed to predict in the future on a network scale. All layers in this autoencoder are trainable. After training the autoencoder for 1000 epochs, the weights matrix of the layer with 200 nodes is extracted. From this matrix, the highest positive weights for each node is kept, and the connected node is identified. All identified nodes are saved and will be used in the following stage of the model. Only positive weights are included because the layers use Relu activation functions which would remove negate the influence of negative weights. Figure 29 illustrates the selected segments to encode the city of Xian.

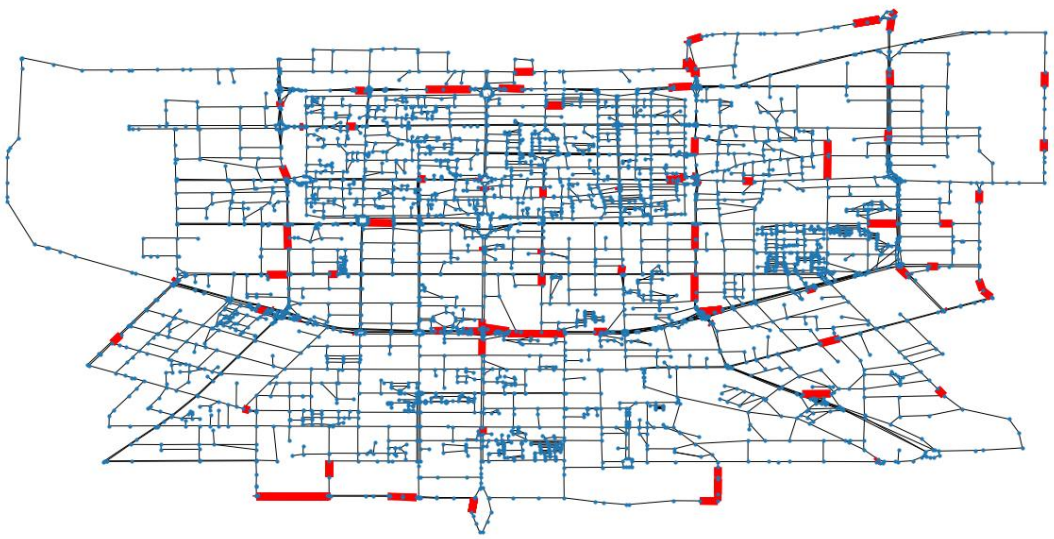


Figure 29: The result of time-autoencoding in Xian, red colour indicates the roads extracted from time-autoencoding

3.5.2 Recurrent model implementation

3.5.2.1 General Idea

The main goal of the model is to use its predictions to predict future steps. This is different from the four-hour multistep model, which uses only fixed inputs. Figure 30 illustrates the process of the recurrent model. The model can be trained for different time horizons. To reduce the complexity of the model, the trainable parameters are shared in each instance of the model. The loss focusses on the overall error sum of all outputs. Using recurrent models that contain recurrent neural networks inside, requires significant computational resources. To reduce the complexity of these models, I reduced the dimension of the tensor passing from the convolution to the recurrent layers by adapting the max-pooling parameters. Why this simplifies the model is explained in section 3.2.2. In this thesis, only prototypes are tested because the computational resources required for efficiently testing these complex models were not available.

Some pre-processing is done after the autoencoding to obtain the inputs in the adjacency matrix structure. The output shape of the model is a vector with the length of all segments used in the input adjacency matrix. The vector needs to be reconstructed into an adjacency matrix before it can be reused. This is done by the 'reconstruction model'. This model has no trainable weights and is defined based on the structure of the output vector and adjacency matrix input.

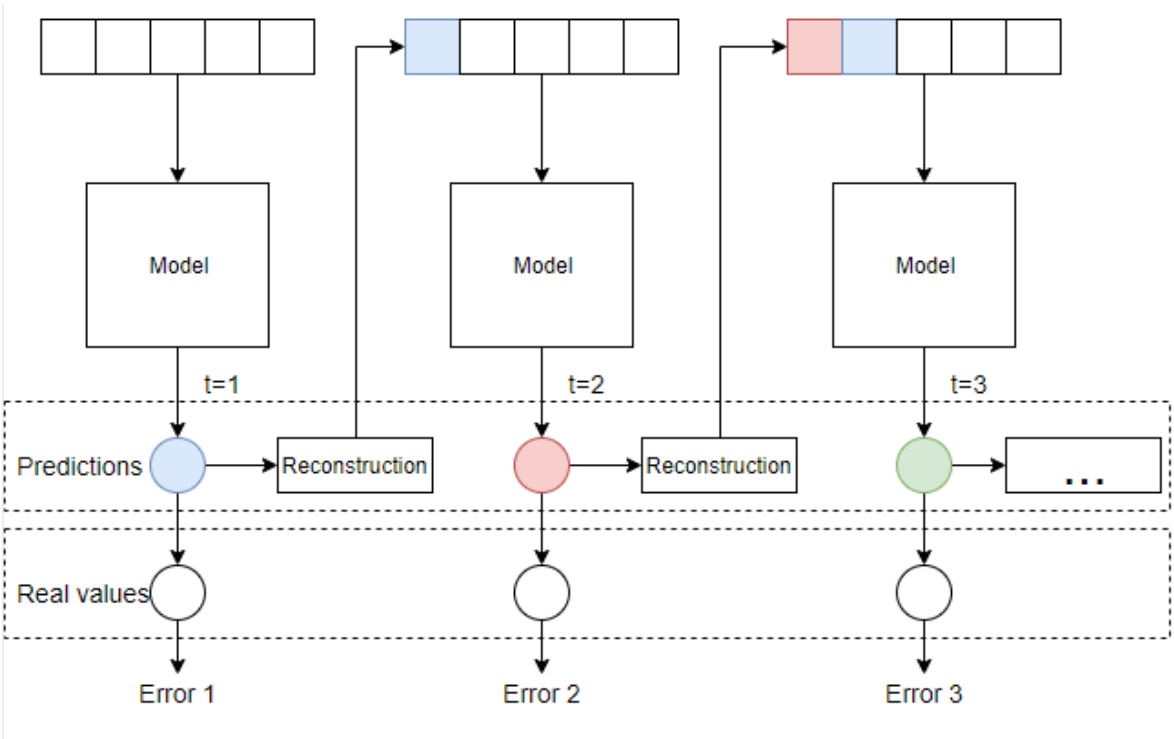


Figure 30: Recurrent model implementation

3.5.2.2 *Implementations*

Taking a closer look at the model, it can be divided into sub-models, as shown in Figure 31. These sub-models each have their functionality. The models need to extract the same spatial and temporal features on each future timestep and thus are the CNN and RNN layers shared between all instances of the model. The FFNN is the last sub-model before the output; these layers combine the features extracted by the CNN-RNN sub-models. Extra inputs can be added to the FFNN to give the model contextual information. This thesis evaluates three prototypes: (i) identical model implementation, (ii) trainable FFNN-layers and (iii) identical model implementation with a special extra input. Firstly, the trainable FFNN layers would allow the model to learn different patterns in the given timesteps, the downfall of this approach is the number of trainable parameters which increases linearly with every new implementation of the model. The special inputs model is an experiment with the same goal as trainable FFNN layers. These special inputs are constant zero tensors locally connected to have a trainable bias, which is then concatenated to the FFNN sub-model inputs. Ideally, the model would learn which extra inputs are needed to have more accurate predictions. Previous experiments have shown that extra inputs like quarter of the day do not suffice as a time measure. The trainable extra inputs might show patterns that were not detected during the manual data analysis.

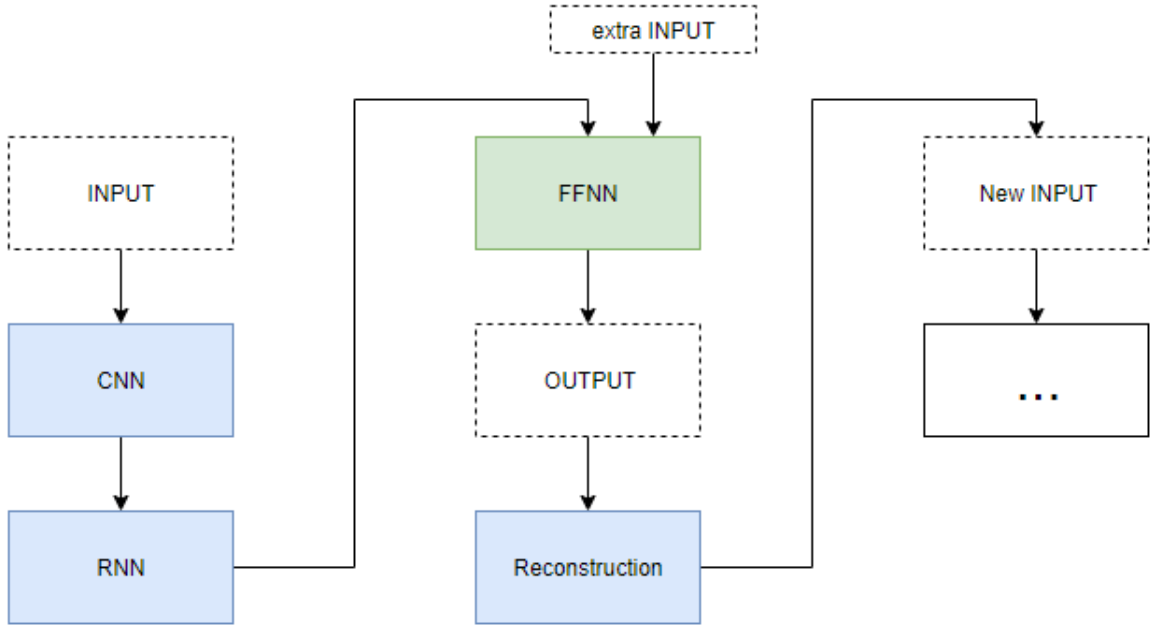


Figure 31: An overview of the model structure

3.5.3 Decoder

The final step in this three-stage model is to decode the predicted values back to a network's scale. This is done with a decoder shown in Figure 32. This decoder has an input tensor which contains the average speed of the segments extracted by the time-autoencoder. The output tensor contains the information of all other road-segments in the traffic network. The decoder is trained on the training set with no time shift.

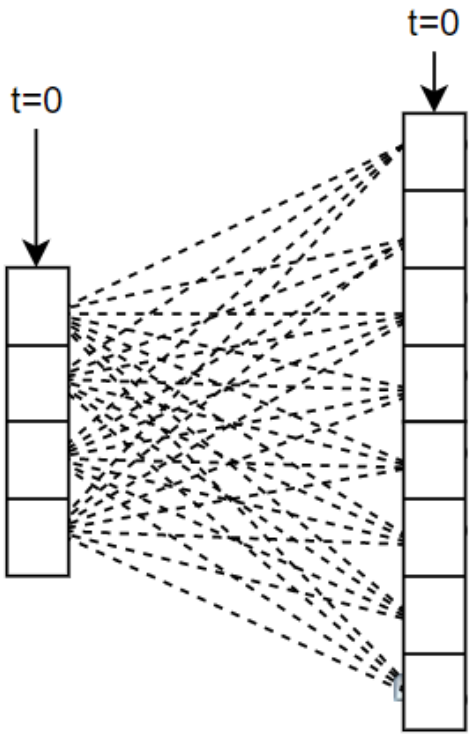


Figure 32: Decoder

4 Results

This section presents the comparison of the proposed pre-processing method, a comparison of the model architectures suggested in this work and finally an evaluation of the network-wide prediction model. First two comparisons will be made for both a five-minute prediction and a multistage four-hour prediction. With the use of sparse GPS data, there is always some noise on the time-series due to using probe vehicles. This complete signal will be considered as ground truth, and the noise will cause a certain portion of an evaluator's error. An improvement will look small percentage wise due to the fixed error of the noise. To ensure the results are valid, a second evaluation is done using the Friedman test (Ranking) with Holm post-hoc method. This second analysis evaluates the randomness in the test results. Friedman's test ranks given methods for each sample; after this ranking, the average of the methods are calculated resulting in the final ranking. Table 7 illustrates this algorithm. The post-hoc method gives an indication of the chance of randomness in the test results. If two algorithms scores are very similar, Friedman's test might overlook this. The post-hoc method gives a percentage of certainty that methods are similar or not. Lower values imply that the methods are significantly different in performance.

Based on RMSE-weighted	Method 1	Method 2	Method 3
Sample 1	1	3	2
Sample 2	2	3	1
Sample 3	1	2	3
Resulting rank	**1.33**	**2.66**	**2**

Table 6: Friedman's test example

4.1 Evaluation of pre-processing

Three pre-processing approaches will be evaluated: in-going and outgoing traffic flow, regular correlation and the suggested time-based correlation. These approaches are tested on all the suggested models to have an overall performance indication. Table 7 shows the average error over all segments and time from the methods mentioned above. From these results, it can be concluded that the spatial feature has lesser influence regarding short-term predictions. For long-term predictions, the importance of this feature rises. Figure 33 shows this trend. Time-based correlation focuses on overall prediction accuracy and outperforms the other methods on every step except for the first two. In these steps, the difference is insignificant, for short term predictions (up to 15 min) information from the segment itself or surrounding segments suffices for an accurate prediction.

5 min	Rank	MAE	MAPE	RMSE
In- Outgoing	2.0417	3.481/2.354/1.228	0.126/0.089/0.051	5.110/4.092/2.704
Correlation	2.0833	3.467/2.347/1.226	0.126/0.088/0.051	5.094/4.080/2.699
Time-Correlation	**1.875**	**3.460/2.340/1.220**	**0.125/0.088/0.050**	**5.093/4.076/2.688**
5h multistep	**Rank**	**MAE**	**MAPE**	**RMSE**
In- Outgoing	2.55	4.192/2.977/1.763	0.148/0.107/0.066	5.969/4.995/3.761
Correlation	1.9167	4.135/2.930/1.724	0.146/0.106/0.065	5.912/4.939/3.700
Time-Correlation	**1.5333**	**4.111/2.909/1.708**	**0.145/0.105/0.065**	**5.879/4.905/3.663**

Table 7: Scoring pre-processing

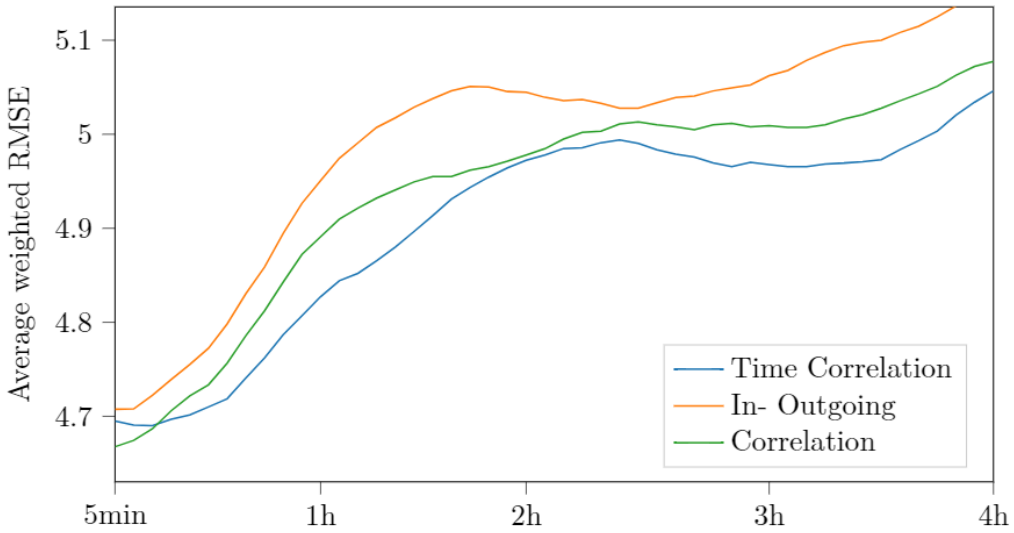


Figure 33: Average weighted RMSE pre-processing

During the post-hoc evaluation, time-based correlation is compared to the other techniques. Table 8 contains the post-hoc values. These values confirm that all methods have similar results for short-term predictions. The value of time-based correlation can be noticed if applied to long-term prediction, the chance that these results are due to randomness is lower than 4%. From these results, the conclusion is made that time-based correlation is significantly better compared to regular correlation and in- outgoing traffic approaches. Further experiments will use the time-based correlation.

Method	5 min – pHolm [%]	4 h – pHolm [%]
Correlation	94.0973	0
In- outgoing	94.0973	3.5764

Table 8: Post-hoc pre-processing

4.2 Evaluation of suggested models

The models that will be evaluated are presented in section 3.3.3. Table 9 presents the scores obtained for all the tested models. The models are tested on all validation segments for both five minutes and four hours predictions. Evaluating the results, Convmax-sigmoid outperforms all other models. Convmax, which is in second place, has the exact same hyperparameters but no sigmoid activation function on the output. This activation contributes to the accuracy of the model, so the hypothesis explained in section 3.4 is most likely correct. The difference in evaluated errors are rather small, but as explained above, there is a fixed error due to the noise of sparse GPS-data, and thus small improvements are more significant than they appear to be. Furthermore, these results insinuate that the influence of proposed contextual information does not help to predict in short-term nor long-term up to 4h.

5 min	Rank	MAE	MAPE	RMSE
Convmax	2.1667	3.433/2.318/1.204	0.124/0.087/0.050	**5.077**/4.062/2.678
Convmax-sigmoid	**1.6667**	**3.430/2.315/1.201**	**0.124/0.087/0.050**	5.078/**4.060/2.667**
Extra-input	3.1667	3.584/2.440/1.295	0.129/0.091/0.053	5.198/4.181/2.811
Extra-input-RNN	4.9167	3.679/2.515/1.350	0.132/0.093/0.055	5.287/4.271/2.913
Maxconv	3.0833	3.527/2.397/1.266	0.127/0.089/0.051	5.148/4.135/2.763
4h multistep	**Rank**	**MAE**	**MAPE**	**RMSE**
Convmax	2.0833	4.046/2.847/1.648	0.144/0.104/0.064	5.791/4.808/3.545
Convmax-sigmoid	**1.75**	**4.034/2.839/1.644**	**0.143/0.103/0.064**	**5.783/4.800/3.537**
Extra-input	4.0833	4.201/2.996/1.790	0.146/0.106/0.066	6.004/5.040/3.823
Extra-input-RNN	4.5833	4.215/3.006/1.797	0.147/0.107/0.066	6.012/5.050/3.836
Maxconv	2.5	4.059/2.862/1.664	0.145/0.105/0.065	5.807/4.830/3.578

Table 9: Performance of suggested models

Convmax-Sigmoid is compared to the other models in Table 10. This post-hoc evaluation confirms the above-mentioned conclusions.

Method	5 min – pHolm [%]	4 h – pHolm [%]
Convmax	43.8578	60.5577
Extra-input	6.041	0.0902
Extra-input-RNN	0.0002	0.0045
Maxconv	6.041	49.0556

Table 10: Post-hoc model evaluation

To analyse what spatial features are extracted during the convolution, a visualisation of the intermediate outputs is shown in Figure 34. To have an understandable illustration Shared vision module MMCCM, or max pooling followed by convolution as described in Table 3, is used. This convolution has fewer parameters and smaller dimensions, which makes it more convenient to visualise. The input to the convolution model is the same adjacency matrix shown in Figure 19 at 12:30 on 1 October 2016. During the first convolution, individual clusters are grouped to form more unified clusters. During the second stage convolution individual pixels or combinations of segments are extracted that are most relevant to the segment to predict for. The last layer uses max pooling to reduce the size of the tensor. In this example, the input to the pooling layer contains mainly the same information as the output, which is a great conversion as the last tensor occupies less memory.

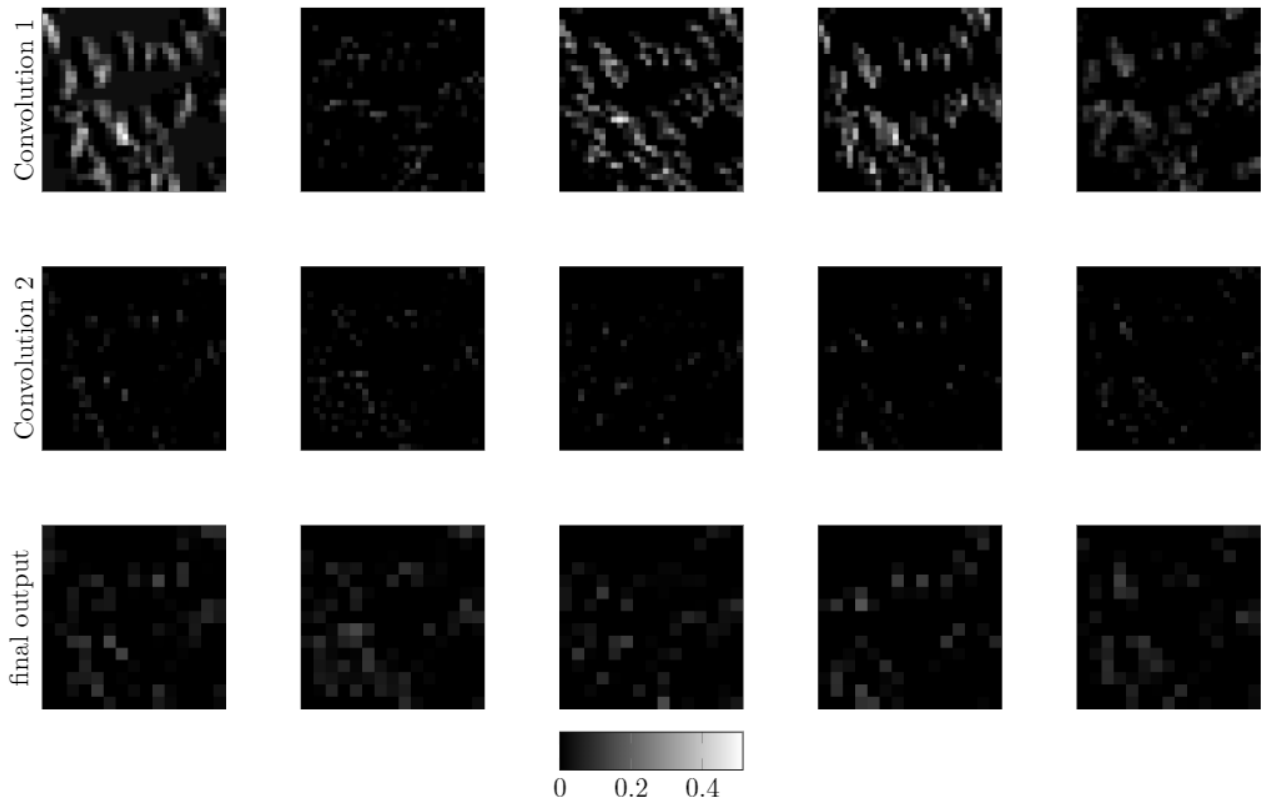


Figure 34: Intermediate outputs of convolution

4.3 Comparison versus state-of-the-art

This section will compare the performance of the suggested best model versus more traditional approaches. More advanced models are evaluated in section 4.4. First, the best performing model Convmax-Sigmoid will be compared to the benchmark models using the same pre-processing. This is followed by a comparison with the benchmark lacking the suggested pre-processing and using the in - outgoing traffic method. The benchmark consists of an LSTM trained with only information of the edge itself, an LSTM, SVM and KNN trained with the same information contained in adjacency matrices. From Table 11, it can be seen that the suggested model outperforms all other benchmarks. The post-hoc analysis resulted in all Holms probability factor under 6% except for the LSTM with identical information, which has 20% of uncertainty. The difference is less if compared to in- outgoing traffic pre-processing applied to the benchmarks. As this is only a five-minute prediction, the time-based pre-processing technique has less of an influence. The Holm percentages are 15% and 36% for LSTM and LSTM-same-info respectively, the other factors are below 3% and thus insignificant. Due to the exploding gradients in both LSTM benchmarks, a numeric analysis would be an unfair comparison. To have the most honest comparison, a graphical comparison is conducted for long-term predictions.

5 min	Ranking	MAE	MAPE	RMSE
Convmax-Sigmoid	**1.4167**	**3.430/2.315/1.201**	**0.124/0.087/0.050**	**5.078/4.060/2.667**
KNN	4.8333	3.797/2.592/1.388	0.139/0.098/0.057	5.511/4.462/3.065
LSTM	2.8333	3.495/2.363/1.231	0.128/0.089/0.051	5.123/4.104/2.720
LSTM-network-wide	2.25	3.479/2.352/1.226	0.127/0.089/0.051	5.099/4.083/2.702
SVM	3.6667	3.653/2.499/1.345	0.132/0.094/0.055	5.239/4.224/2.862
4h multistep	**Ranking**	**MAE**	**MAPE**	**RMSE**
Convmax-Sigmoid	**1.25**	**4.034/2.839/1.644**	**0.143/0.103/0.064**	**5.783/4.800/3.537**
KNN	3.8333	4.439/3.193/1.947	0.158/0.116/0.073	6.348/5.387/4.190
LSTM	4.9167	5.296/3.703/2.111	inf/inf/inf	6.972/5.784/4.260
LSTM-network-wide	2.4167	4.381/3.076/1.770	inf/inf/inf	6.078/5.040/3.704
SVM	2.5833	4.355/3.089/1.822	0.157/0.114/0.070	6.052/5.053/3.782

Table 11: Evaluation versus benchmark with pre-processing

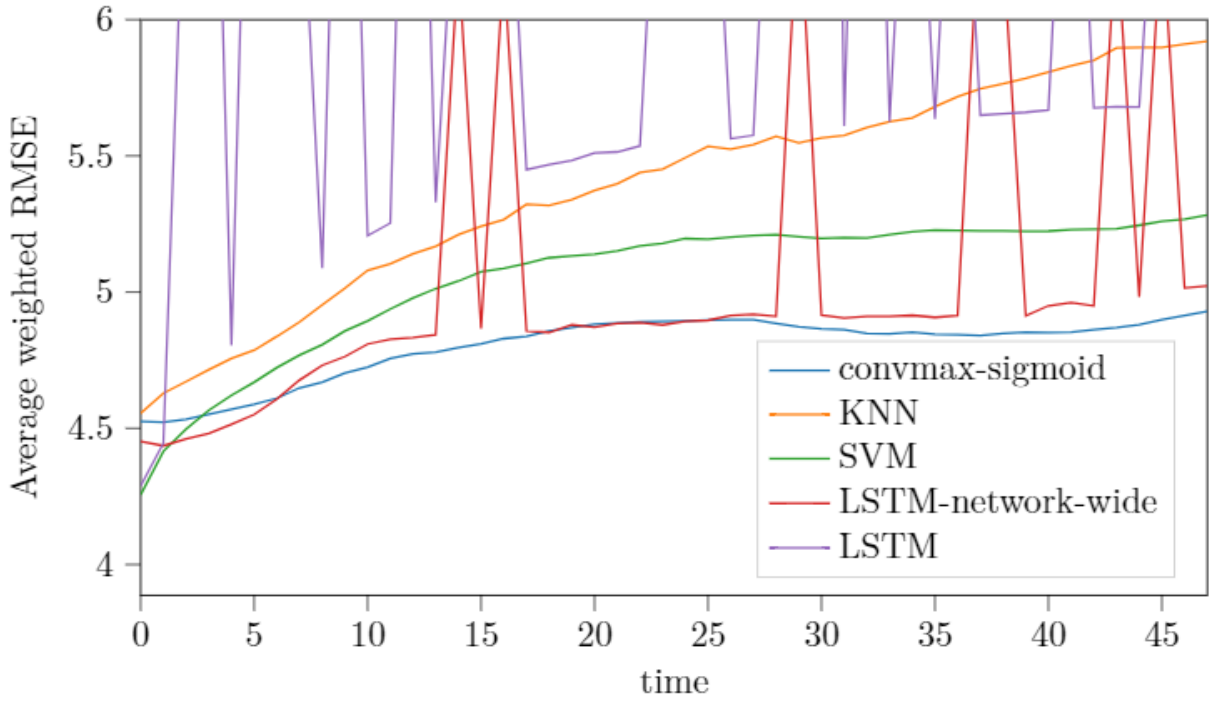


Figure 35: Average weighted RMSE benchmark with pre-processing

Figure 35 represents the benchmark models with time-based correlation compared to the Convmax-Sigmoid model using time-based correlation. It can be noticed that the exploding gradients cause peaks in the average weighted RMSE. Ignoring these peaks and comparing the model to the benchmarks, it can be concluded that our model outperforms the benchmarks. The LSTM which is trained with the same information as the model is closest in performance. There is still a small gap between the two graphs; this gap is due to the spatial component, which is present in our model but not in the LSTM. Convmax-Sigmoid has the most consistent performance compared to the other methods. Below 15-minute predictions, the model has a lesser performance. This can be explained with the manner of training and pre-processing. Convmax-Sigmoid is trained to have consistent performance over the complete time-horizon. If trained for short-term predictions, the model outperforms all other algorithms as shown in Table 11.

5 min	Ranking	MAE	MAPE	RMSE
Convmax-Sigmoid	**1.5833**	**3.430/2.315/1.201**	**0.124/0.087/0.050**	**5.078/4.060/2.667**
KNN	4.9167	3.801/2.601/1.400	0.139/0.098/0.058	5.534/4.483/3.082
LSTM	2.5833	3.496/2.363/1.230	0.129/0.090/0.051	5.124/4.105/2.721
LSTM-network-wide	2.1667	3.486/2.356/1.227	0.128/0.089/0.051	5.104/4.087/2.702
SVM	3.75	3.672/2.514/1.356	0.133/0.094/0.056	5.243/4.228/2.867
4h multistep	**Ranking**	**MAE**	**MAPE**	**RMSE**
Convmax-Sigmoid	**1**	**4.034/2.839/1.644**	**0.143/0.103/0.064**	**5.783/4.800/3.537**
KNN	3.833	4.566/3.265/1.963	0.164/0.119/0.075	6.440/5.442/4.197
LSTM	4.9167	5.546/3.872/2.197	inf/inf/inf	7.212/5.978/4.394
LSTM-network-wide	2.6667	4.550/3.188/1.827	inf/inf/inf	6.235/5.172/3.807
SVM	2.5833	4.451/3.148/1.846	0.160/0.115/0.070	6.144/5.133/3.849

Table 12: Evaluation versus benchmark without pre-processing

Figure 36 represent Convmax-Sigmoid again using time-based correlation but is now compared to the benchmark using traditional in- outgoing traffic pre-processing. The overall performance of the benchmarks decreases and thus shows the value of time-based pre-processing.

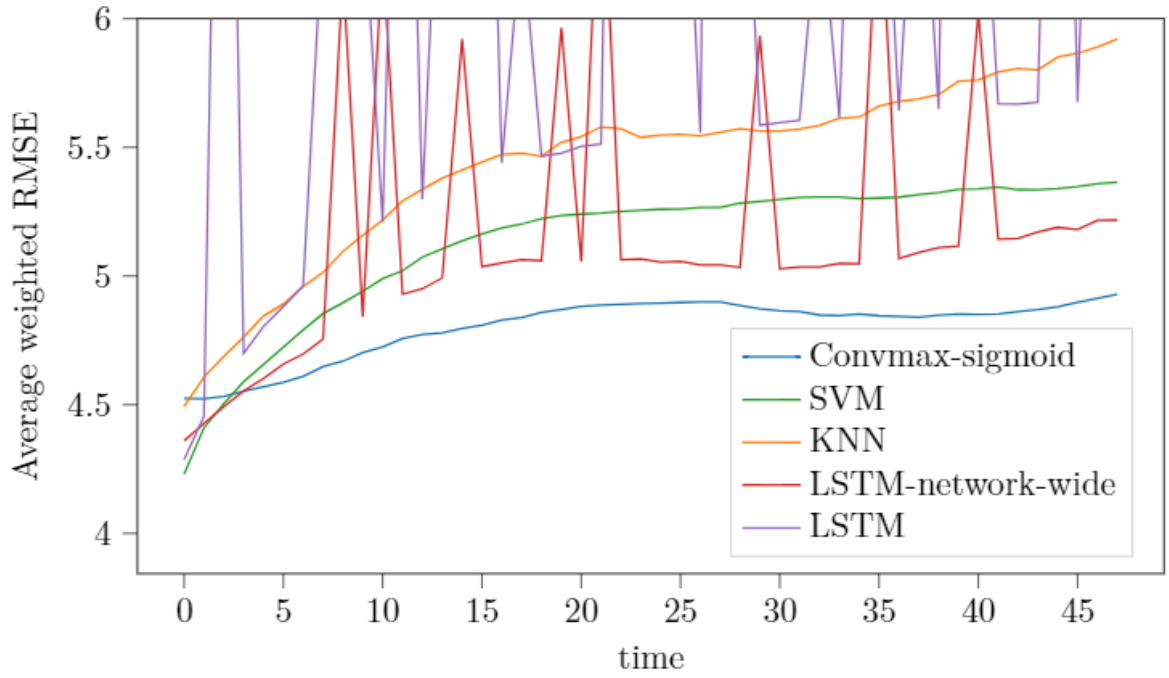


Figure 36: Average weighted RMSE benchmark without pre-processing

4.4 Analysis of TRANSFOR 19 forecasting competition

This transportation forecasting competition was organised by ABJ70 Artificial intelligence and advanced computing applications, supported by IEEE ITSS technical activities sub-committee 'Smart Cities and Smart Mobility' and sponsored by DiDi Chuxing. The idea was for participants to develop a traffic forecasting algorithm predicting the average speed of a road every five minutes. This road had two directions and predictions should be made between 6.00-11.00 and 16.00-21.00 on the final day of the DiDi data set. All GPS-traces flowing through this road between the mentioned time interval were removed. Participants had one month to develop the algorithm. The average RMSE is evaluated during the first round of the competition, and the best five algorithms proceed to the next round.

Figure 37 shows the top 6 classified in round one[7]. In the second round, the scoring of the participants was based on the accuracy (50%), novelty (25%), quality of the code (10%) and a presentation at the TRB 2019 Annual Meeting conference (15%)[8]. The proposal done by the University of Deusto scored fifth in the first round, ending up in third place during the second round. Given the promising results, we realised some programming mistakes and neglecting the removed data on the prediction day, which drastically lowered our performance. With our streamlined models, the RMSE for both segments improved from 5.11 to 2.35 north (segment 2748) and 4.3596 to 2.36 south (segment 160) resulting in an overall score of 2.36. This new RMSE for both segments is shown in Figure 38 and Figure 39, together with a 4h prediction coming from the multistep model. We expanded our model to predict for four hours in multistep to show the potential of the suggested model.

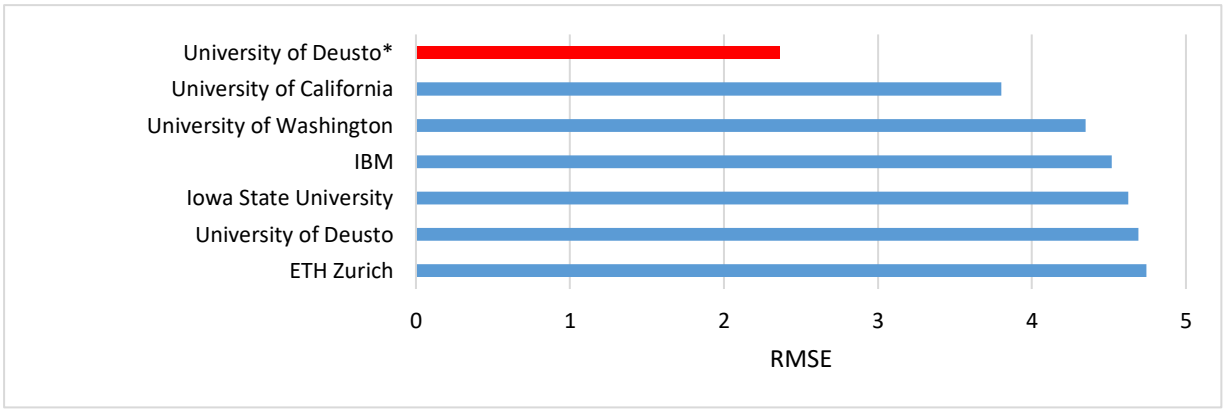


Figure 37: Scoring RMSE TRANSFOR19

[7] https://github.com/TRANSFORABJ70/TRANSFOR19

[8] https://sites.google.com/site/trbcommitteeabj70/abj70s-latest-news/another-successful-trb-for-abj70?authuser=0

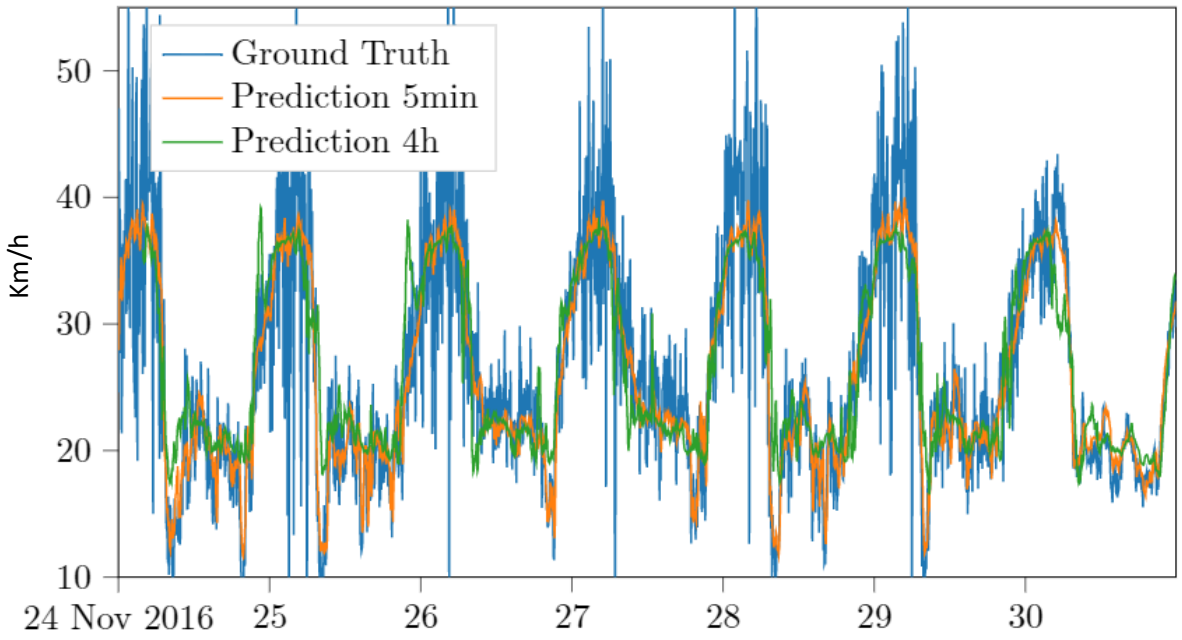
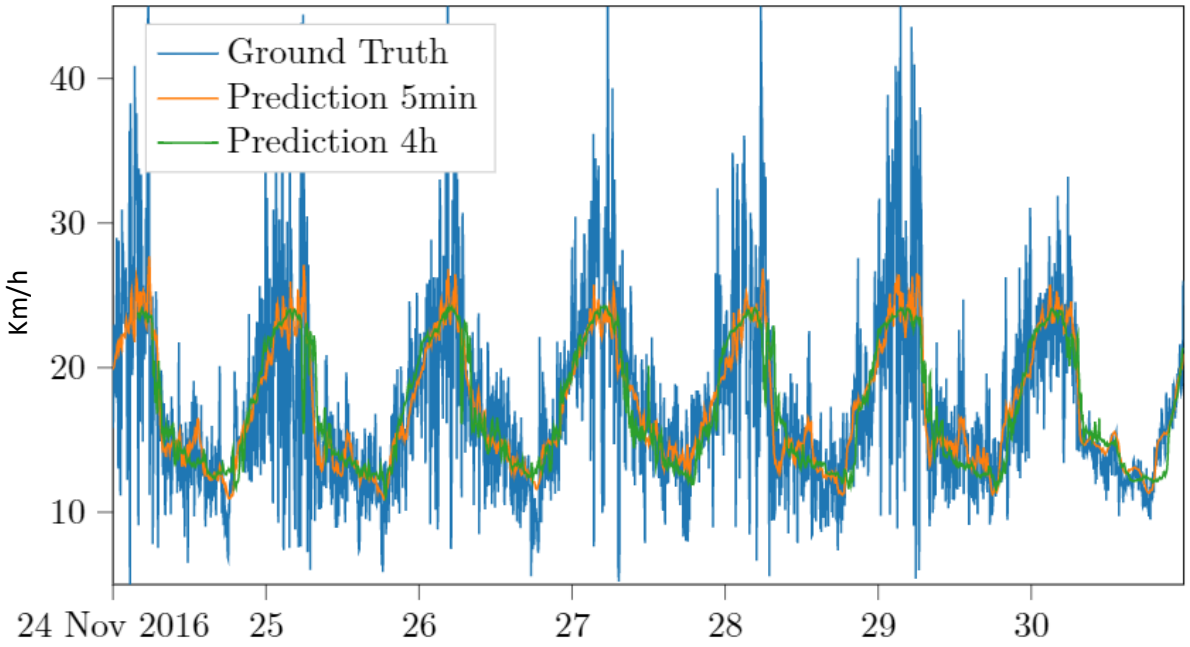


Figure 38: Predictions Segment 2748 (south segment)



1010

Figure 39: Predictions segment 160 (north segment)

4.5 Evaluation of suggested city-wide recurrent model

In this section, the results of the models designed for network-wide predictions will be evaluated. Firstly Section 4.5.1 will analyse the results based on five-minute predictions on a network scale. This is followed by the analysis for longer time-horizon prediction on a network-wide scale using recurrent models in Section 4.5.2.

4.5.1 Five-minute predictions

The first advancement compared to previous models is to extend predictions to a network-wide scale. The three-stage-model which will be evaluated, consists of:

- Time-autoencoding
- Recurrent model, a single implementation
- Decoding

A detailed explanation of the model can be found in Section 3.5. The three stages are trained individually. Time-autoencoding extracts the segments that will encode the city's traffic situation. The input of the auto-encoder is filtered to remove segments with high levels of missing data and/or untrustworthy data. Once the encoding segments are found, the adjacency matrices are constructed. The following two stages, being the model implementation and decoding, are trained individually to reduce computational resources required. The decoder is trained with real values of the encoding segments to predict the real values of decoded segments. It would be beneficial to combine the training process of the model and decoder. This would allow the decoder to compensate any errors made during predictions of the model.

As mentioned above, the models tested are only prototypes and are not yet refined to optimise performance. To test the suggested model, the cities network is reduced to only contain primary and secondary road segments. The total number of segments in this reduced network is 2121. The Number of segments to encode the city's traffic situation is 187 segments (see Figure 29). This number is chosen as a trade-off between efficiency and data-coverage; fewer segments will improve efficiency while more segments will result in more information being passed down and thus better accuracy.

To evaluate network-wide predictions, the average weighted RMSE over the testing set (last week of data) is considered. Figure 40 illustrates the results obtained after the full process of encoding, training the model and decoding.

Figure 40: Averaged weighted RMSE test set Xian

At first glance, the predictions look accurate across the entire city. There are two extremes present, being the top left corner with a perfect fit and the lower left with the highest error values. This could be explained with the availability of data. Ride-hauling GPS-traces are used, these traces are denser inside the city centre compared to the outskirts. The extremes occur on the outskirts of the city where the population coverage of local segments is less compared to the city centre. Before analysing the relationship between missing data and resulting RMSE, a more in-depth evaluation of the predictions is conducted. Figure 41 illustrates a more detailed look at the predicted traffic network. It can be seen just how many segments are contained in a small region.

Figure 41: Zoomed section average weighted RMSE network-wide prediction

The total averaged weighted RMSE is 4.5531. To give some meaning to these numbers, Figure 42 illustrates the average and extremes as well as the distribution of the weighted RMSE. In this figure, A represents the histogram of the weighted RMSE. The bin with the most samples centres around 4.2. B gives an example of an average prediction; this section is decoded thus is not contained in the encoded 187 segments. C Illustrates the prediction for the segment with the highest error value. This segments data is extremely noisy due to a missing factor of 97%. This means only 3% of timesteps have real recorded values. Due to the missing value imputation algorithm, the noise is spread across other timesteps resulting in overall noisy data, which is extremely difficult to predict. At last, D represents a segment with a perfect prediction. This perfect prediction results from 99.9% missing values. The imputation method's last resort is to extrapolate the remaining values after trying to utilise imputation. If there are so little measurements, this will result in a constant line. The decoder consists of a single fully connected layer; this layer has weights and biases. While analysing the parameter matrix of this layer, the weights are negative or zero, and the bias at the node, representing the segment, equals the constant value of the output. This will result in a perfect fit independent of the input. Some more statistics of the network-wide predictions are given in Table 13. The averaged windowed RMSE is 3.247, this value is lower than the best performing models of the TRANSFOR competition. During the competition, predictions are made on only two segments, sometimes using different models for each segment to achieve the best fit. The designed model can averagely outperform, on a network's scale, advanced algorithms designed to only predict individual roads.

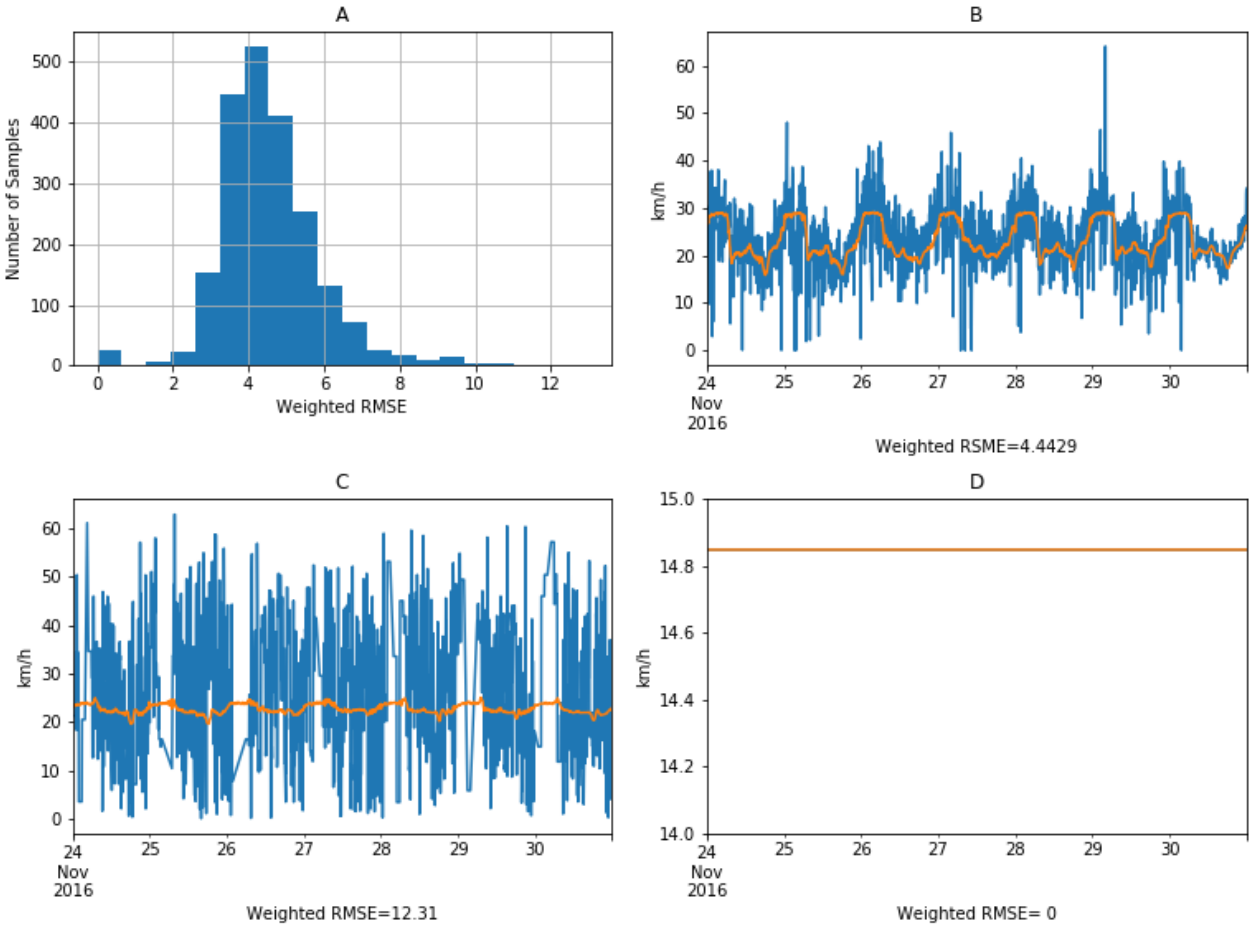


Figure 42: Network-wide prediction description, Blue: ground-truth, Orange: predicted values

Evaluator	[regular / weighted / windowed]
Average RMSE	5.557 / 4.553 / 3.247
Average MAPE	0.226 / 0.161 / 0.096
Average MAE	4.160 / 2.874 / 1.587
Average RMSE decoded segments	5.595 / 4.587 / 3.273
Average RMSE encoded segments	5.162 / 4.197 / 2.921

Table 13: Statistics network-wide prediction

1075

Figure 43 illustrates the relationship between missing values and corresponding weighted RMSE. This figure illustrates that the model can make an accurate prediction of data with less than 30% missing data. If more data is missing, the corresponding error increases. This trend ends at a missing percentage value of 95%. In this region, the models corresponding errors are widely distributed. This distribution can be explained with the model being able to learn to imputation method and making accurate predictions. Although high error values can also be obtained if the imputation algorithm has a noisy input resulting in overall noisy data as explained above.

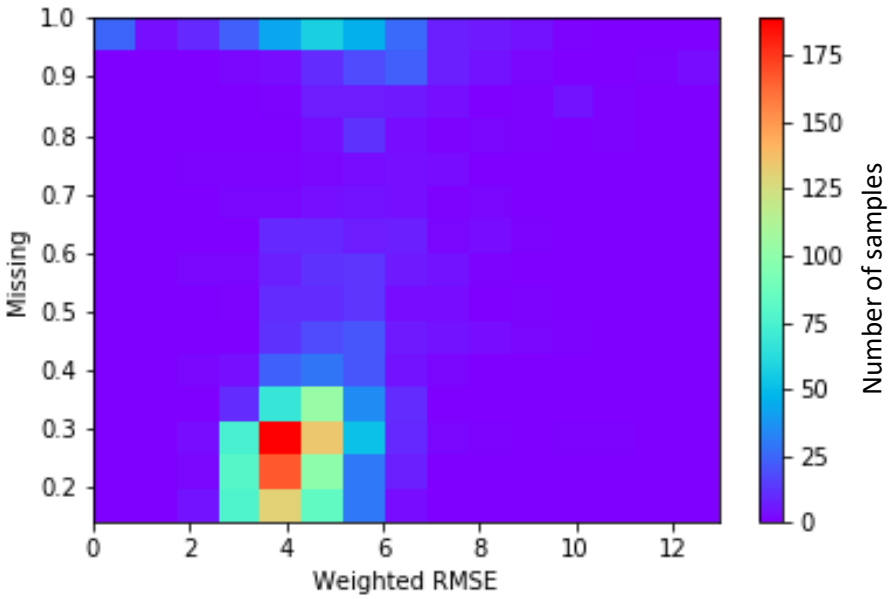


Figure 43: Heatmap missing percentage vs RMSE

4.5.2 Long-term predictions

The models tested in this section are either trained for 36 or 12 future timesteps. Three versions of the same model will be evaluated being:

- Recurrent model with full weight sharing
- Recurrent model with trainable FFNN layers
- Recurrent model with full weight sharing but with trainable extra inputs.

A more detailed explanation of these model can be found in Section 3.5.2.2.

If the models are trained for only 12 timesteps or 1-hour prediction horizon, the performance is acceptable. The performance at five minutes or timestep 0, is comparable to the performance of the previous model used in network-wide predictions. The inclination of the RMSE increase is rather low, as shown in Figure 44. Within this figure, 'Trainable FFNN' means the model uses trainable FFNN's, 'recurrent model' is the regular model, and 'recurrent model-extra' represents the model using trainable extra inputs.

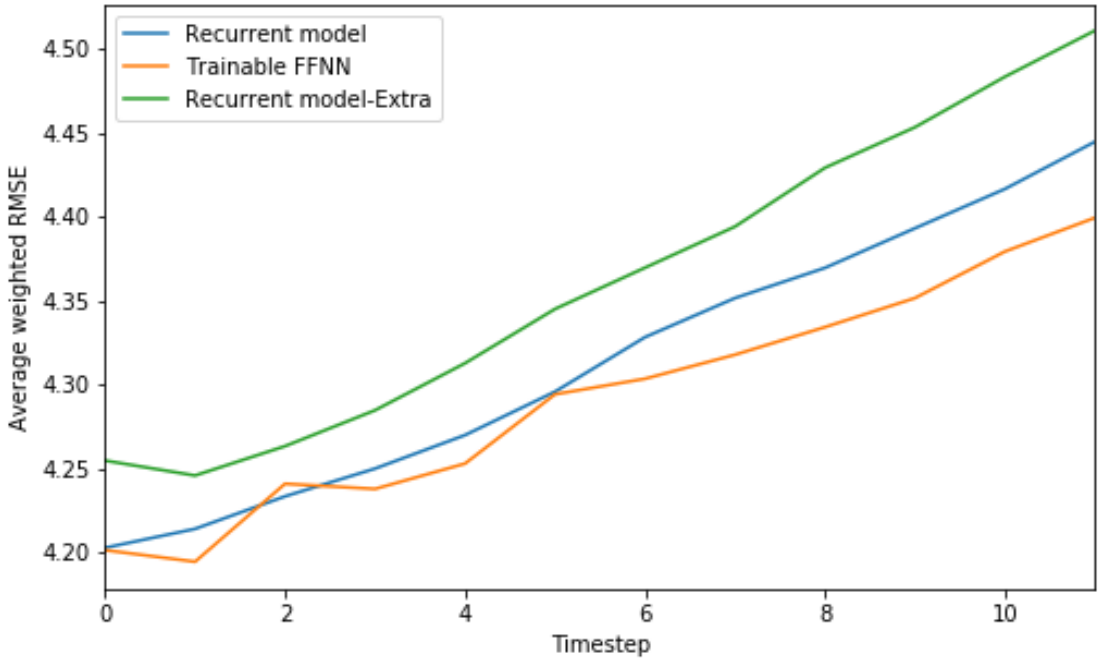


Figure 44: Average weighted RMSE, encoded segments trained for 12 timesteps

Figure 45 illustrates example predictions of encoded segments using the general recurrent model. A trend which occurs in both predictions is the 'squaring' of the prediction. This trend is more visible if the model is trained for more timesteps, as shown in Figure 46. To reduce this 'squaring' effect, a noise layer is introduced before the reconstruction module in the model. The idea of introducing noise is to adapt the prediction values to match more the pattern of the real values. This solution had no significant impact but is kept in the model as it also helps to reduce overfitting.

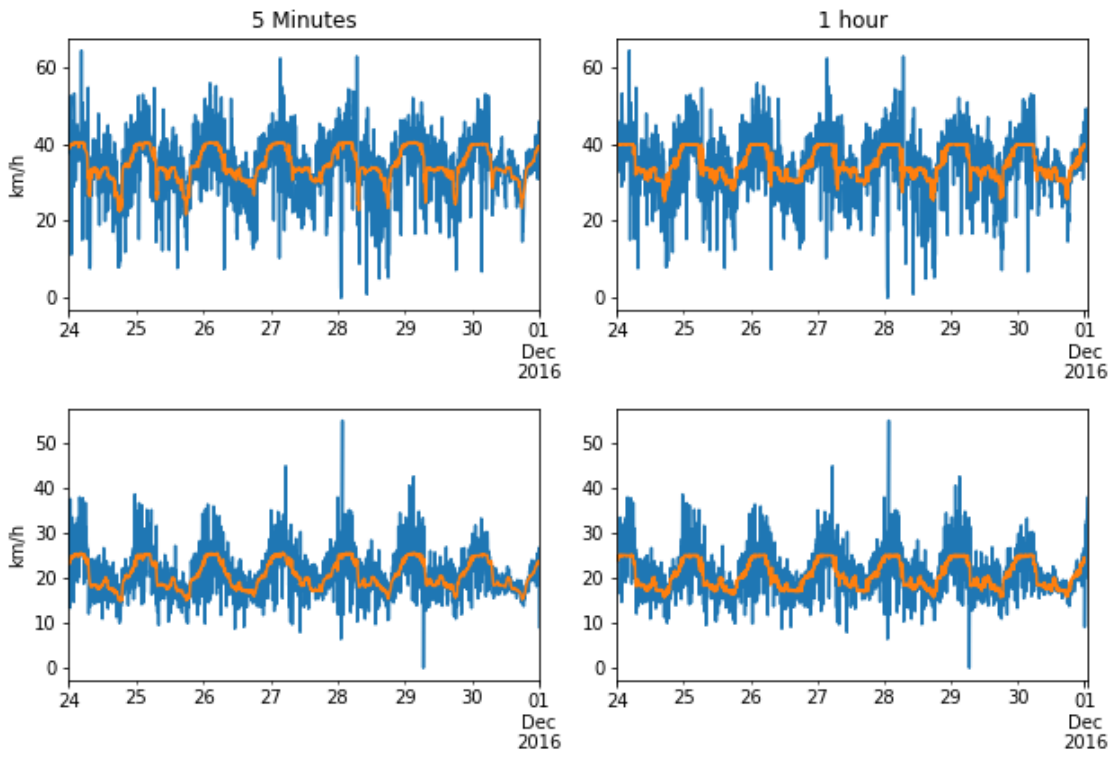


Figure 45: Example predictions recurrent model trained with time-horizon 1h

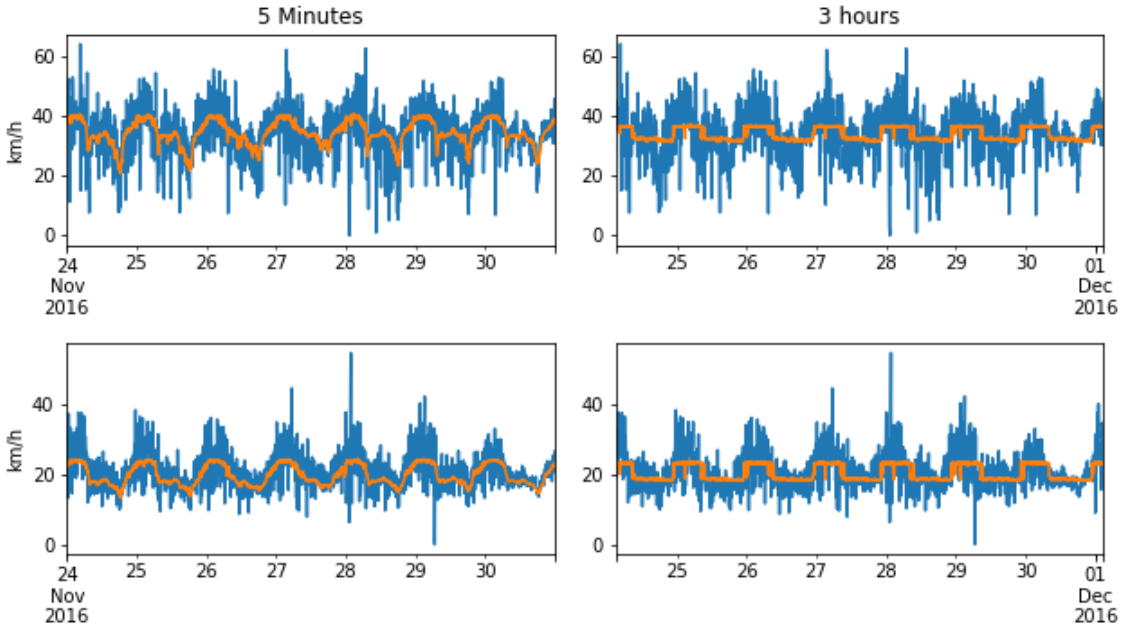


Figure 46: Example predictions recurrent model trained with time-horizon 3h

Two variant models which are based on the general model are developed to try and reduce the 'squaring' effect.

- The first variant is 'Trainable FFNN'; this method allows the model to learn some parameters in every new instance. The downfall of this method is the linear rise of parameters in relation to the time-horizon. The regular model has a constant 3 596 597 parameters for all horizons. The Trainable FFNN model has 6 695 392 parameters if trained for a three-hour horizon.
- The second approach is an ambitious approach to let the model generate its own input. The idea is to let the model create its own extra inputs to 'remember' or make slight changes to the main algorithm on certain timesteps.

Figure 47 illustrates the average weighted RMSE over different time steps. The Trainable FFNN model has the best performance for each step. However, these models still experience the squaring effect, as shown in Figure 48 and Figure 49. The Special Inputs model does not even recognise trends and only learned the most fitting constant predictions. These predictions are created using the bias on the last layer with small influences of the weights. This prediction changes slightly over time due to the influence of the previous prediction and the propagated error resulting in a rise of RMSE over time.

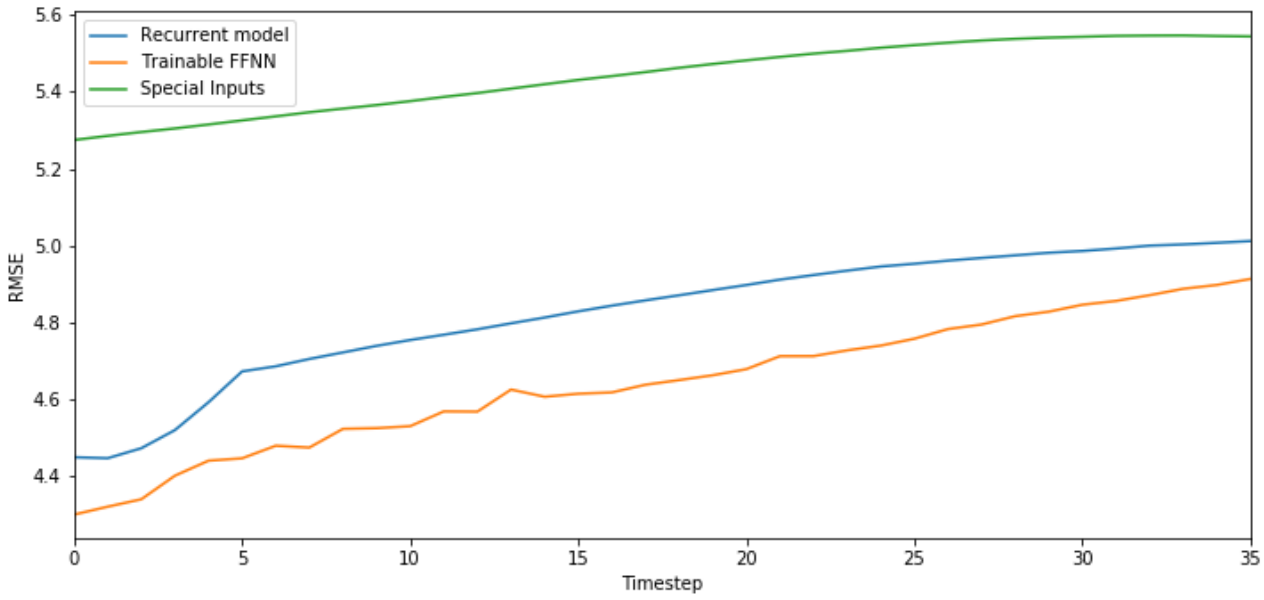


Figure 47: Average weighted RMSE, encoded segments trained for 36 timesteps

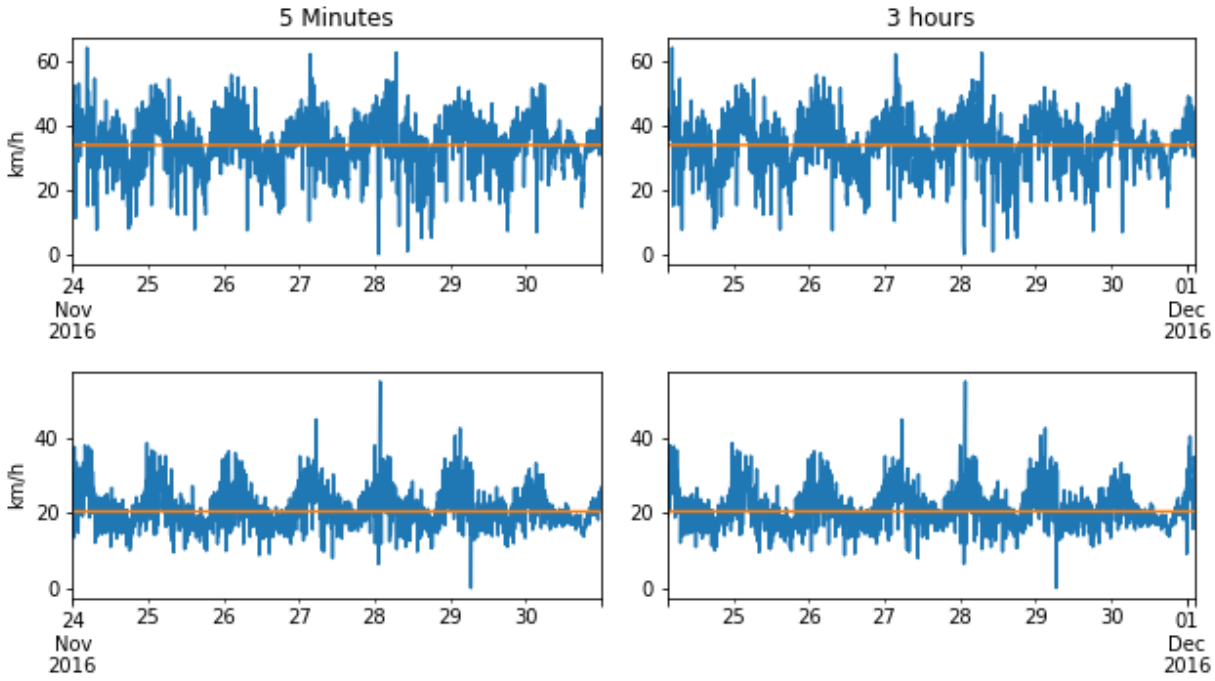


Figure 48: Example predictions Special inputs model trained with time-horizon 3h

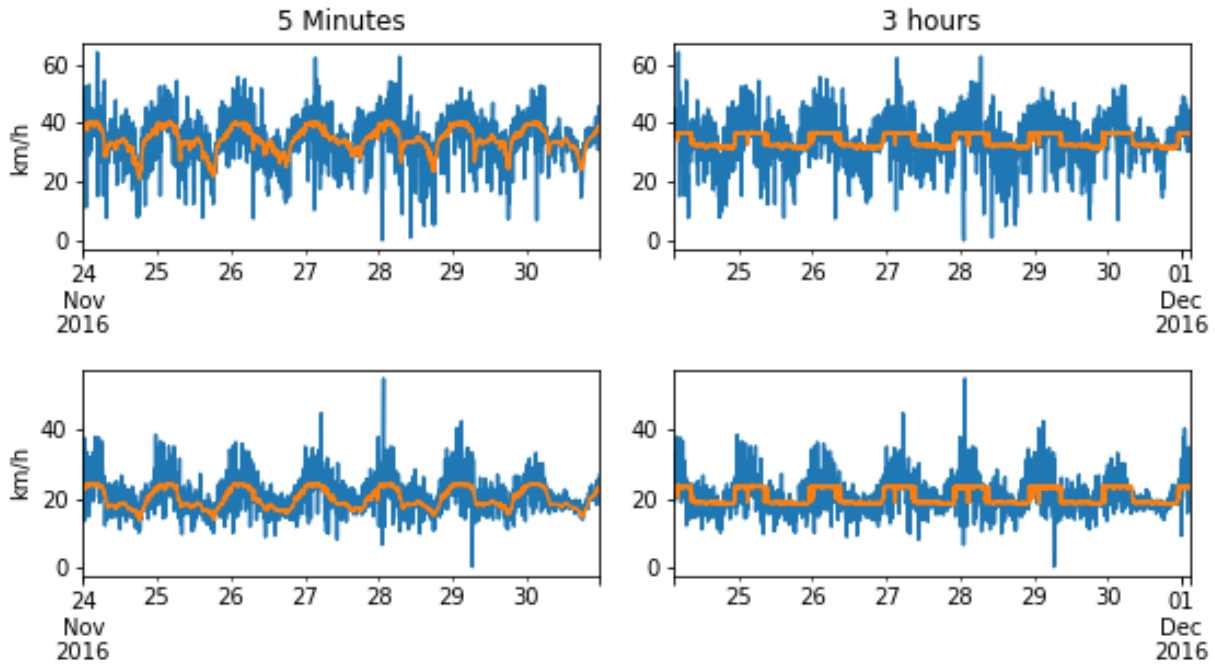


Figure 49: Example predictions Trainable FFNN model trained with time-horizon 3h

Due to this 'squaring' effect, the models are not further tested to decode for network-wide predictions. The origin of this squaring effect is unknown but might be linked to the following aspects:

- Error propagation: Recurrent models reuse the same weights and bias over all implementations. Using the bias within the weighted sum is an easier solution for the optimiser compared to changing the weights. A small change in weights propagates over all timesteps and thus as a high influence further in the chain. The Special Inputs model is an example of pure bias learning resulting in constant predictions. Figure 50 illustrates this trend where the optimizer chooses for these constant predictions as local optimum. The Recurrent model (Blue) starts 'learning' from epoch 35, before this epoch it had a constant value solution.
- Underfitting: The model is unable to learn all the trends present in the data for long-term predictions. Underfitting can be compensated by using more complex/larger neural networks. Another solution can be to extend learning. The learning curves illustrate that the models are still learning close to epoch 100. Extending the number of epoch and thus, training time could improve performance.

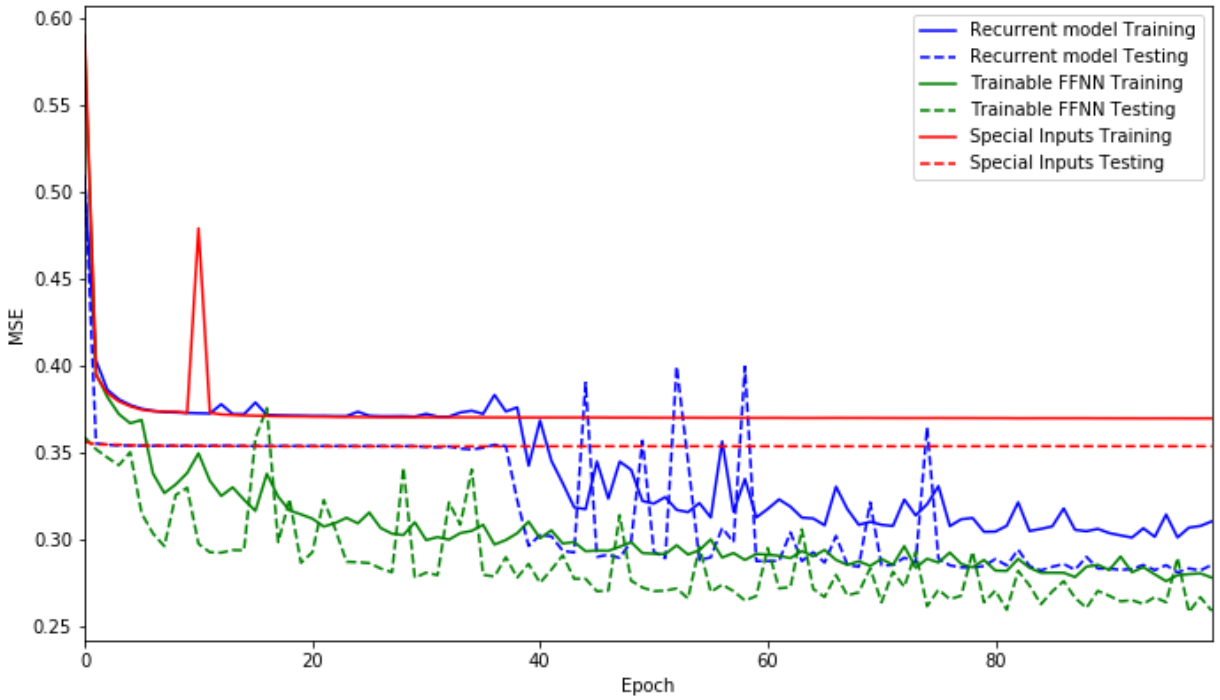


Figure 50: Learning curves

5 Discussion

The main contributions of this thesis are distributed over different aspects of traffic forecasting. This chapter will briefly elaborate on each aspect.

5.1 Pre-processing

Pre-processing is an important step in TF. According to the state-of-the-art, current DL methods face the following challenges: model training time, required computational resources and scalability of models.

5.1.1 Time-correlation

An efficient data reduction technique is developed for graph structures. This reduction can be used to extract the *n* most relevant links, in relation to the link of interest, from a network-wide structure. The reduction uses the time-related correlation of the previous *d* samples to map the influence of each segment compared to the reference segment.

This technique can be applied to both short-term predictions and long-term multistep predictions. In both cases, the technique outperformed other strategies such as regular correlation or ingoing/outgoing traffic flow selections.

5.1.2 Time-autoencoder

This reduction technique is developed to encode a city's traffic situation. The encoder extracts *d* links based on their *n* previous timesteps to reconstruct the following timestep on a network's scale. This time-autoencoder is the first part of the three-stage model used for network-wide predictions.

5.2 Activation function in traffic forecasting

It is uncommon to use activation functions on the output layers of regression applications. However when predicting traffic flow, the range to predict is known between zero and the maximum allowed speed on a link. The Sigmoid function is applied to the output, which results in a better distribution of the predictions and thus improves accuracy.

5.3 Architecture

A hybrid (DNN) architecture for TF was developed. The architecture can find both spatial and temporal relations in traffic data coming from GPS traces. The hybrid DNN consisted of three main components:

(i) Graph-convolutional neural network

(ii) LSTM neural network

(iii) Feedforward neural network

5.3.1 Single link predictions

The model can be used to predict for a single link of a network. The predictions range from five minutes (short-term) up-to four hours in multi-step (long-term). The best performing model for both short-term and long-term was the suggested Convmax-Sigmoid model.

5.3.1.1 Short-term

The Convmax-Sigmoid model was compared to regular benchmarks being LSTM, KNN and SVM. The proposed model outperformed all benchmarks. When the model is combined with the suggest time-correlation and benchmarks use regular pre-processing techniques, the performance difference was more significant. An initial model was used in the TRANSFOR19 competition ranking 3rd. This initial model contained certain mistakes; after refining the model, the performance surpassed the best model of the forecasting competition with a significant margin.

5.3.1.2 Long-term

To explore the capabilities of the hybrid model, the prediction horizon was expanded to four hours. For long-term predictions, the influence of the suggested time-correlation became visible. The model was able to keep a high performance over the entire time horizon. When comparing the model to benchmarks using regular pre-processing techniques, the suggested architecture shows a statistically significant improvement.

5.3.2 Network-wide predictions

A three-stage model is designed to allow for network-wide predictions. This model consists of three main blocks:

(i) Time-autoencoder
(ii) Recurrent model implementation
(iii) Decoder

The recurrent model architecture is based on the architecture of the Convmax-Sigmoid model with some alterations.

5.3.2.1 Short-term

The model can efficiently make accurate predictions on a network-wide scale. The average error over the entire network is below the error of the best performing model, which only predicts for two links, used in the TRANSFOR19 competition. Network-wide prediction is a significant challenge in the state-of-the-art. Very little literature addresses this subject. The suggested approach is an efficient way to address this problem keeping high performance over the entire network.

5.3.2.2 Long-term

The recurrent models were able to make accurate predictions when used on a time-horizon up-to 1h. Increasing the time horizon resulted in 'squaring' of the predictions. Due to this trend, the models were not further tested on a network-wide scale. Variants of the model were designed to reduce this trend but were unsuccessful. In section 7, suggestions are made to solve the 'squaring' effect.

6 Conclusion

In this work, a hybrid (DNN) architecture is proposed for single link predictions as well as network-wide predictions regarding TF.

Firstly, the single link model was able to find both spatial and temporal relations in traffic data coming from GPS traces. The hybrid DNN consisted of three main components:

(i) Graph-convolutional neural network,

(ii) LSTM neural network

(iii) Feedforward neural network

The proposed GraphCNN-LSTM architecture was able to predict traffic short-term (5 minutes) time horizons and long-term time horizons using multistep predictions (up to 4 hours). Furthermore, a data reduction technique is proposed that selects the *n* most relevant links from a city-wide road network using time-related correlation.

The proposals were tested using ride-hauling GPS-data from the cities of Xi'an and Chengdu, two big cities in China. The data is provided by the DiDi Chuxing Gaia Open Data Initiative; this data was used during the TRANSFOR19 forecasting competition. The main contributions of the proposal are as followed.

- The data reduction technique based on time-related correlation performs better than data reductions methods based only on correlation or upstream- and downstream-road links
- The variant of our proposal based on Convmax-sigmoid was the one that obtained the best results in terms of RMSE, MAE and MAPE. However, the difference with respect to the Maxconv was not significant. The latter has significantly less trainable parameters; its use must be appropriate in some scenarios in which the computational resources for training the model are limited.
- The data reduction technique based on time-related correlation allowed to also improve the performance of the baseline algorithms, that is, K-NN, SVM and LSTM.
- Our proposal outperformed baseline algorithms with and without using the data reduction technique and for both short-term and long-term time horizons. Furthermore, in long-term predictions, the difference in accuracy between the proposed model and the baseline methods increased as the time horizon increased.

The network-wide approach was designed to predict on a network-wide scale. This approach consisted of three main parts:

(i) Time-autoencoding
(ii) Recurrent model
(iii) Decoder

The model is tested on the same data as the single link architectures.

The proposed model was able to predict for a short-term time horizon on a network-wide scale with high accuracy. The best performance was recorded on segments with less than 30% missing values. The average city-wide performance surpassed all advanced algorithms used during the TRANSFOR19 forecasting competition. If applied to long-term predictions, the model starts showing flaws for time horizons above one hour. In my opinion, these flaws are related to underfitting and error propagation inside the recurrent models. Suggestions to tackle these flaws are presented in section 7.

To sum up, these methods can provide accurate predictions on a large scale to users or administrative instances, in order to help plan trips, improve mobility, improve efficiency, etc. The models can make these predictions with high accuracy using low computational resources. Largely deployed services like Waze can utilise these methods to anticipate congestion and proactively prevent them. The final aim of ITS is to reduce emission and congestion while improving the efficiency of our transportation systems.

7 Future work

This final chapter will elaborate on future works and planned experiments. In my opinion, this line of research is promising to be further examined. The rise of data collection and the growing interest in ITS keep this topic relevant for the coming years.

Zero propagation in convolution on sparse matrices

When using convolution on sparse matrices such as adjacency matrices, it is possible to identify zero values that have a 100% chance to flow through the entire convolution unit, as shown in Figure 34. Within this figure, the final output has a lot of black squares representing zeros. The experiments hypothesis is that these zeros can be removed while maintaining performance. When removing these zeros, the number of trainable parameters reduces and thus efficiency goes up.

Filling of space in adjacency matrices

This experiment would test the influence of the filler values used in the adjacency matrices. It could be confusing for the model if mostly zeros, which correspond to 'no movement', are used to represent average velocity. This experiment would invert the filling to ones corresponding to 'free flow'.

Pre-trained convolution models

Within the research of computer vision, advanced models are created to tackle complex models. These models could be used as the convolution part of our models.

Transfer learning

Instead of starting with no random weights, trainable parameters can be transferred. As most of the suggested models share similar structures, the weights they use can be initialised based on previous instances.

Longer timesteps

When predicting for long-term time horizons, the five-minute interval becomes less important. There should be a separation between a model trained for five-minute intervals up to one hour and long-term predictions using 30 minutes intervals for horizons longer than 1 hour.

Solving the 'squaring' effect

During the analysis of the long-term network-wide predictions, the squaring of predictions occurred. The experiment would allow the model to learn for more epochs while also combining the training of the decoder and recurrent model. This would allow the decoder to compensate for any errors made by the recurrent models.

A second experiment related to this issue would be to use more complex models. For example, the recurrent models could pass information directly with a hidden state. The recurrent model would be built comparable to current GRU's or LSTM's

Non-recurrent events

This final issue is a common challenge in TF. Sadly, the dataset used did not contain these types of events. As our predictions are based in function of more than 100 different segments, the hypothesis is that the suggested models are more resilient to non-recurrent events. If a non-recurrent event such as a big accident occurs, the distortion will quickly be identified. The model should be able to anticipate the influence based on the influence observed on the related segments.

8 References

[1] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, “Short-term traffic forecasting: Where we are and where we’re going,” *Transp. Res. Part C Emerg. Technol.*, vol. 43, pp. 3–19, 2014.

[2] B. S. Kerner, “The physics of traffic,” *Phys. World*, vol. 12, no. 8, pp. 25–30, Aug. 1999.

[3] M. G. Karlaftis and E. I. Vlahogianni, “Statistical methods versus neural networks in transportation research: Differences, similarities and some insights,” *Transp. Res. Part C Emerg. Technol.*, vol. 19, no. 3, pp. 387–399, 2011.

[4] M. M. Hamed, H. R. Al-Masaeid, and Z. M. B. Said, “Short-term prediction of traffic volume in urban arterials,” *J. Transp. Eng.*, vol. 121, no. 3, pp. 249–254, 1995.

[5] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni, “Road traffic forecasting: recent advances and new challenges,” *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 2, pp. 93–109, 2018.

[6] S. Yang, “On feature selection for traffic congestion prediction,” *Transp. Res. Part C Emerg. Technol.*, vol. 26, pp. 160–169, 2013.

[7] L. Zhao *et al.*, “T-GCN: A Temporal Graph ConvolutionalNetwork for Traffic Prediction,” Nov. 2018.

[8] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, “DNN-based prediction model for spatio-temporal data,” in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2016, p. 92.

[9] A. Ermagun and D. Levinson, “Spatiotemporal traffic forecasting: review and proposed directions,” *Transp. Rev.*, vol. 38, no. 6, pp. 786–814, 2018.

[10] J. S. Angarita-Zapata, A. D. Masegosa, and I. Triguero, “A Taxonomy of Traffic Forecasting Regression Problems from a Supervised Learning Perspective,” *IEEE Access*, p. 1, 2019.

[11] L. Do, N. Taherifar, and H. L. Vu, “Survey of neural network-based models for short-term traffic state prediction,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 9, p. e1285, 2018.

[12] W. Huang, G. Song, H. Hong, and K. Xie, “Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, 2014.

[13] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, “Traffic flow prediction with big data: a deep learning approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, 2014.

[14] Z. Cui, R. Ke, and Y. Wang, “Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction,” *CoRR*, vol. abs/1801.0, 2018.

[15] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, “Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks,” in *Sensors*, 2017.

[16] X. Ma, H. Yu, Y. Wang, and Y. Wang, “Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory,” *PLoS One*, vol. 10, no. 3, pp. 1–17, 2015.

[17] J. Zhang, Y. Zheng, and D. Qi, “Deep Spatio-temporal Residual Networks for Citywide Crowd Flows Prediction,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 1655–1661.

[18] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[19] J. Atwood and D. Towsley, “Diffusion-convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[20] W. Jin, Y. Lin, Z. Wu, and H. Wan, “Spatio-Temporal Recurrent Convolutional Networks for Citywide Short-term Crowd Flows Prediction,” in *Proceedings of the 2Nd International Conference on Compute and Data Analysis*, 2018, pp. 28–35.

[21] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, “High-Order Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting,” 2018.

[22] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting,” *CoRR*, vol. abs/1709.0, 2017.

[23] X. Wei, J. Li, Q. Yuan, K. Chen, A. Zhou, and F. Yang, “Predicting Fine-Grained Traffic Conditions via Spatio-Temporal LSTM,” *Wirel. Commun. Mob. Comput.*, vol. 2019, pp. 1–12, 2019.

[24] S. Howell, “Meta-analysis of machine learning approaches to short-term urban traffic prediction,” in *Scottish Transport Applications and Research Conference*, 2018, pp. 1–15.

[25] J. S. Angarita-Zapata, I. Triguero, and A. D. Masegosa, “A Preliminary Study on Automatic Algorithm Selection for Short-Term Traffic Forecasting,” in *Intelligent Distributed Computing XII*, 2018, pp. 204–214.

[26] T. H. H. Aldhyani and M. R. Joshi, “Clustering to Enhance Network Traffic Forecasting,” in *Information and Communication Technology for Sustainable Development*, 2018, pp. 357–364.

[27] D. Pavlyuk, “Short-term Traffic Forecasting Using Multivariate Autoregressive Models,” *Procedia Eng.*, vol. 178, pp. 57–66, 2017.

[28] J. Brownlee, *Introduction to Time Series Forecasting With Python*, V1.6., vol. 1. Copyright 2018 Jason Brownlee, 2018.

[29] Q. Shi and M. Abdel-Aty, “Big Data applications in real-time traffic operation and safety monitoring and improvement on urban expressways,” *Transp. Res. Part C Emerg. Technol.*, vol. 58, pp. 380–394, 2015.

[30] I. Triguero, G. P. Figueredo, M. Mesgarpour, J. M. Garibaldi, and R. I. John, “Vehicle Incident Hot Spots Identification: An Approach for Big Data,” in *IEEE Trustcom/BigDataSE/ICESS*, 2017, pp. 901–908.

[31] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, “Traffic Flow Prediction With Big Data: A Deep Learning Approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[32] N. G. Polson and V. Sokolov, “Deep learning for short-term traffic flow prediction,” *Transp. Res. Part C Emerg. Technol.*, vol. 79, pp. 1–17, 2017.

[33] Z. Duan, Y. Yang, K. Zhang, Y. Ni, and S. Bajgain, “Improved Deep Hybrid Networks for Urban Traffic Flow Prediction Using Trajectory Data,” *IEEE Access*, vol. 6, pp. 31820–31827, 2018.

[34] Q. Liu, B. Wang, and Y. Zhu, “Short-Term Traffic Speed Forecasting Based on Attention Convolutional Neural Network for Arterials,” *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 11, pp. 999–1016, 2018.

[35] P. Newson and J. Krumm, “Hidden Markov Map Matching Through Noise and Sparseness,” in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009, pp. 336–343.

[36] G. D. Forney, “The viterbi algorithm,” *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[37] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral Networks and Locally Connected Networks on Graphs,” Dec. 2013.

[38] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning Convolutional Neural Networks for Graphs,” 2016.

[39] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” 2013.

9 Appendix

9.1 Workplan

Title

The Master thesis's title changed from 'Machine learning for traffic and logistic problems' to 'Temporal Graph Convolutional Deep Neural Networks for Traffic Forecasting using GPS Data'. First, the master thesis had an open scope to allow for broad research topics. After the traffic forecasting competition, the designed model became the main topic.

Introduction

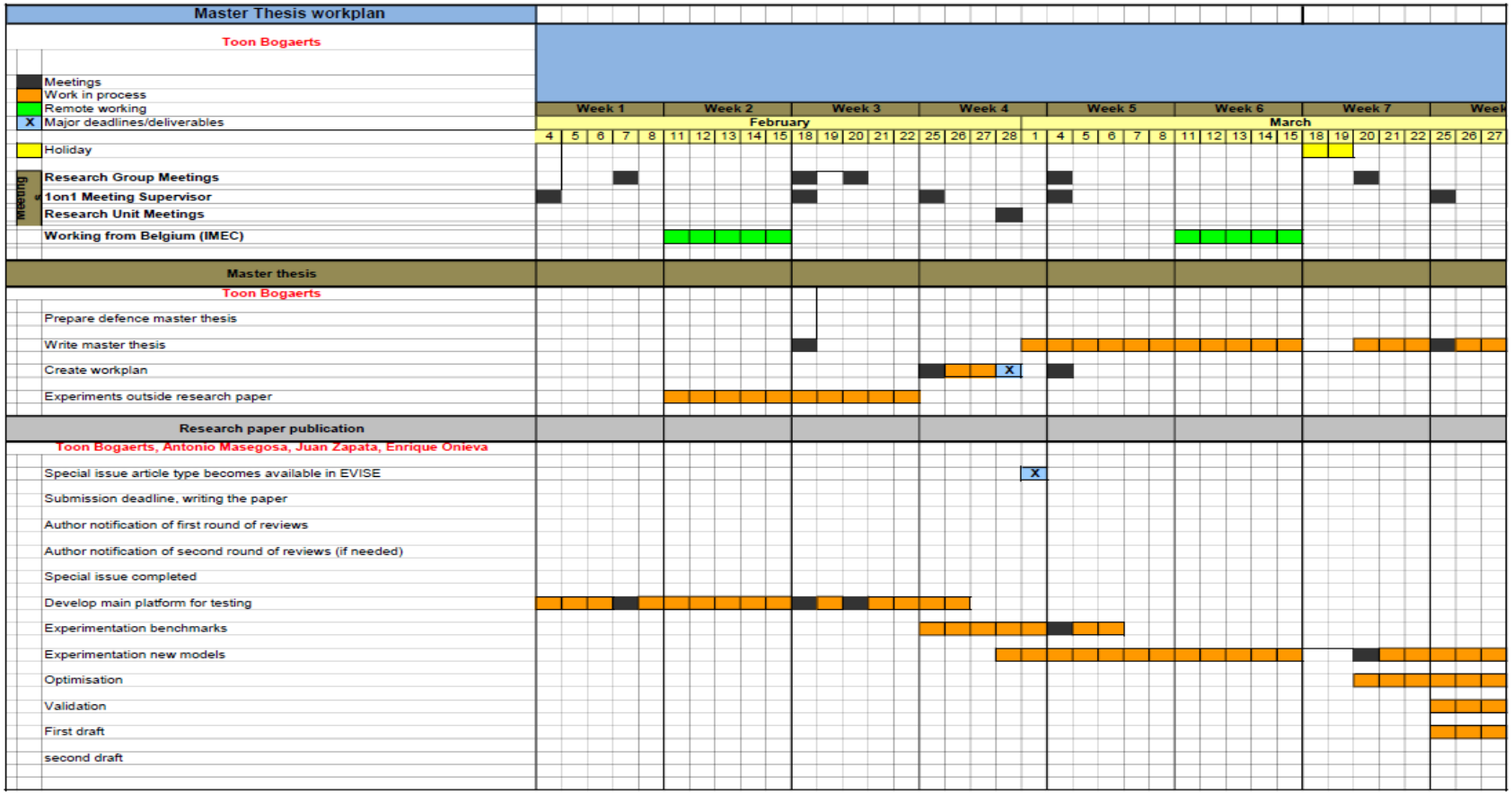
The project is based around forecasting traffic. These forecasts are done in different manners and applied to real GPS-traced trajectories.

Aim

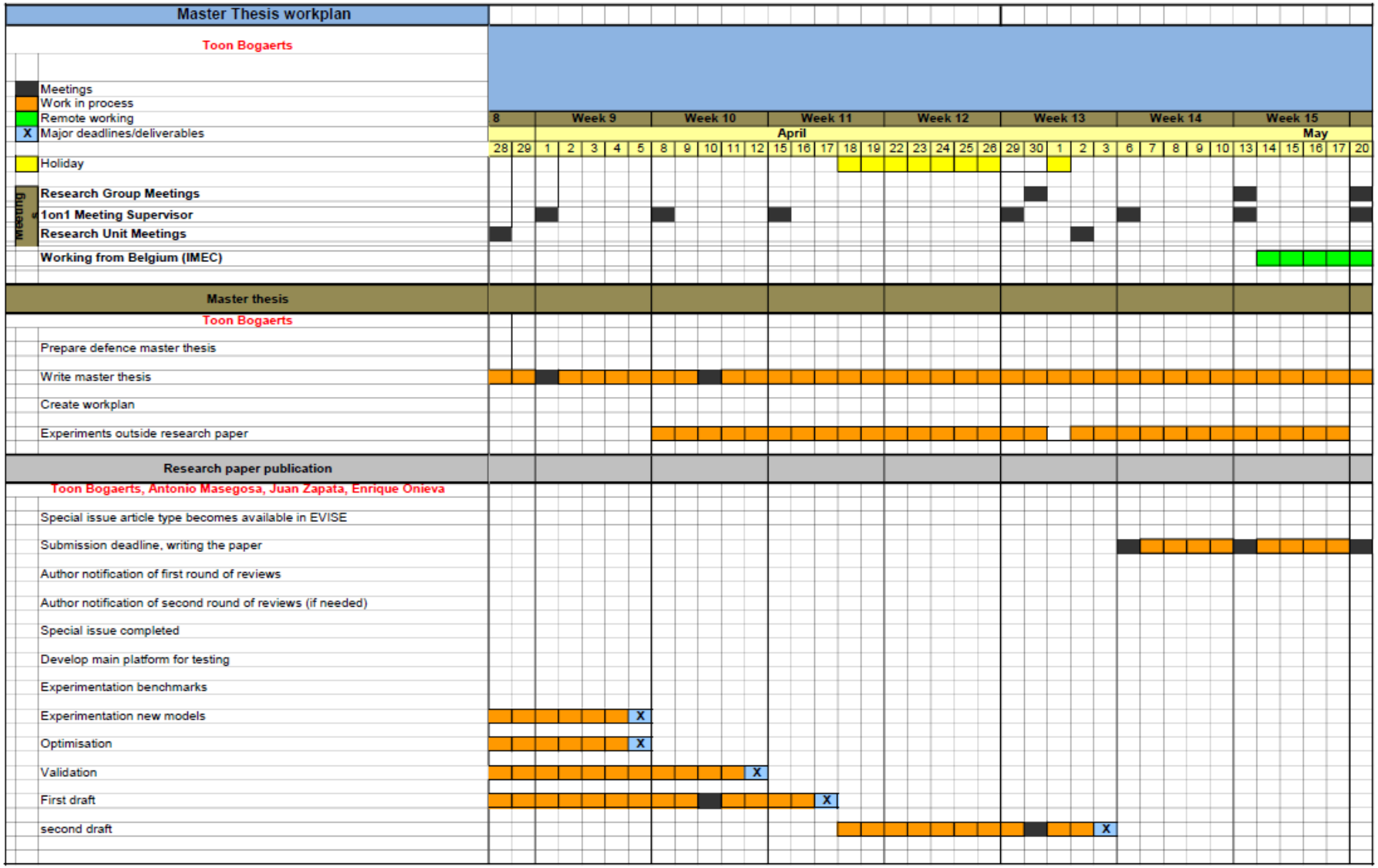
The project aims to optimise and refine the model designed for traffic forecasting. This is followed by exploring the full potential of these models. There were no predefined milestones or objectives; the process of exploring the full potential and optimisation resulted in numerous individual hurdles.

Project plan

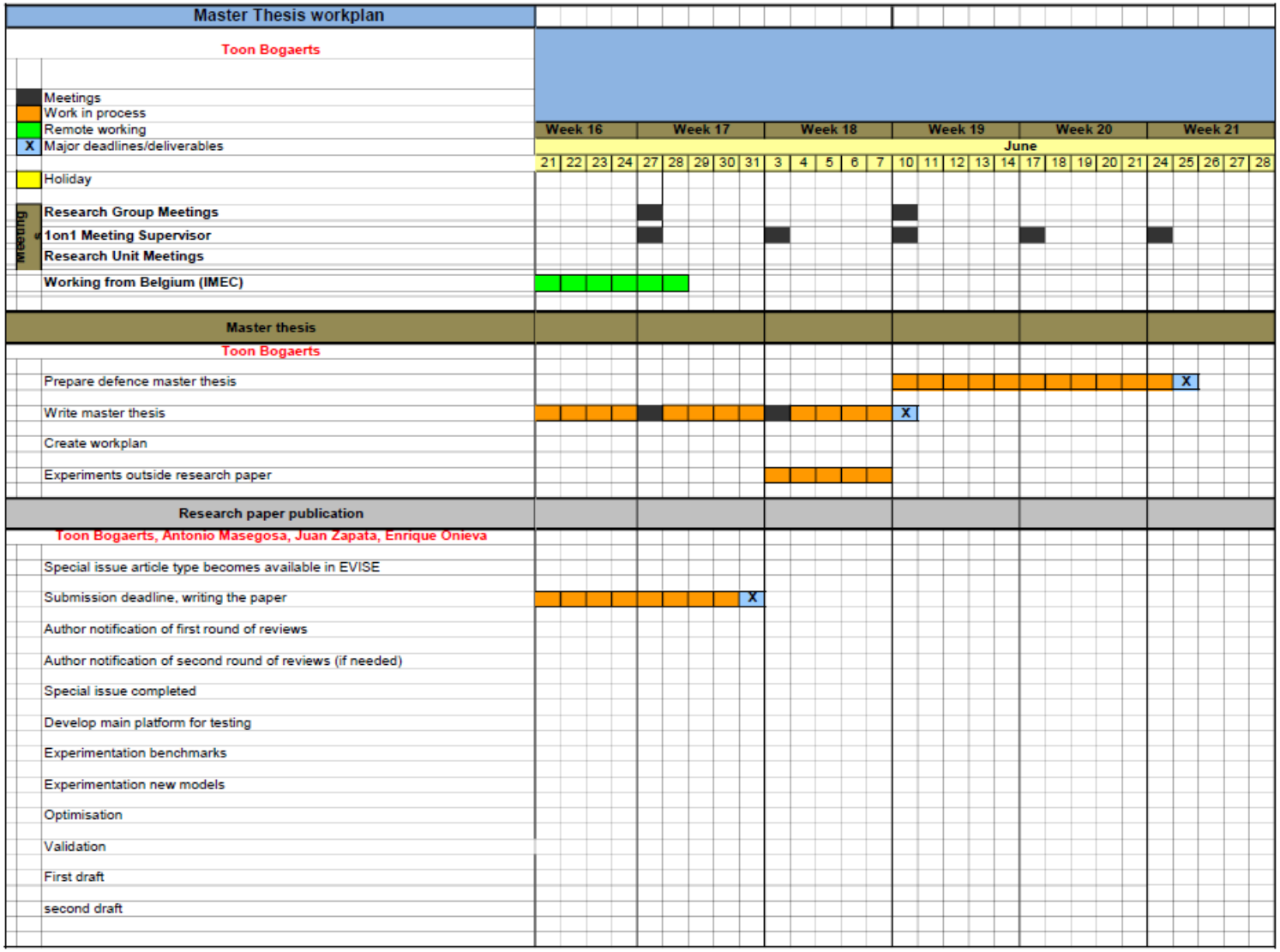
The following pages present the time chart used during this thesis.


Master Thesis workplan
Toon Bogaerts
Meetings
Work in process
Remote working
X Major deadlines/deliverables
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
February
March
4 5 6 7 8 11 12 13 14 15 18 19 20 21 22 25 26 27 28 1 4 5 6 7 8 11 12 13 14 15 18 19 20 21 22 25 26 27
Holiday
Meeting
Research Group Meetings
1on1 Meeting Supervisor
Research Unit Meetings
Working from Belgium (IMEC)
Master thesis
Toon Bogaerts
Prepare defence master thesis
Write master thesis
Create workplan
Experiments outside research paper
Research paper publication
Toon Bogaerts, Antonio Masegosa, Juan Zapata, Enrique Onieva
Special issue article type becomes available in EVISE
Submission deadline, writing the paper
Author notification of first round of reviews
Author notification of second round of reviews (if needed)
Special issue completed
Develop main platform for testing
Experimentation benchmarks
Experimentation new models
Optimisation
Validation
First draft
second draft

1410



Universiteit Antwerpen · Loughborough University · Deusto Universidad de Deusto University of Deusto · université de BORDEAUX



Universiteit Antwerpen · Loughborough University · Deusto Universidad de Deusto University of Deusto · université de BORDEAUX

9.2 Logbook

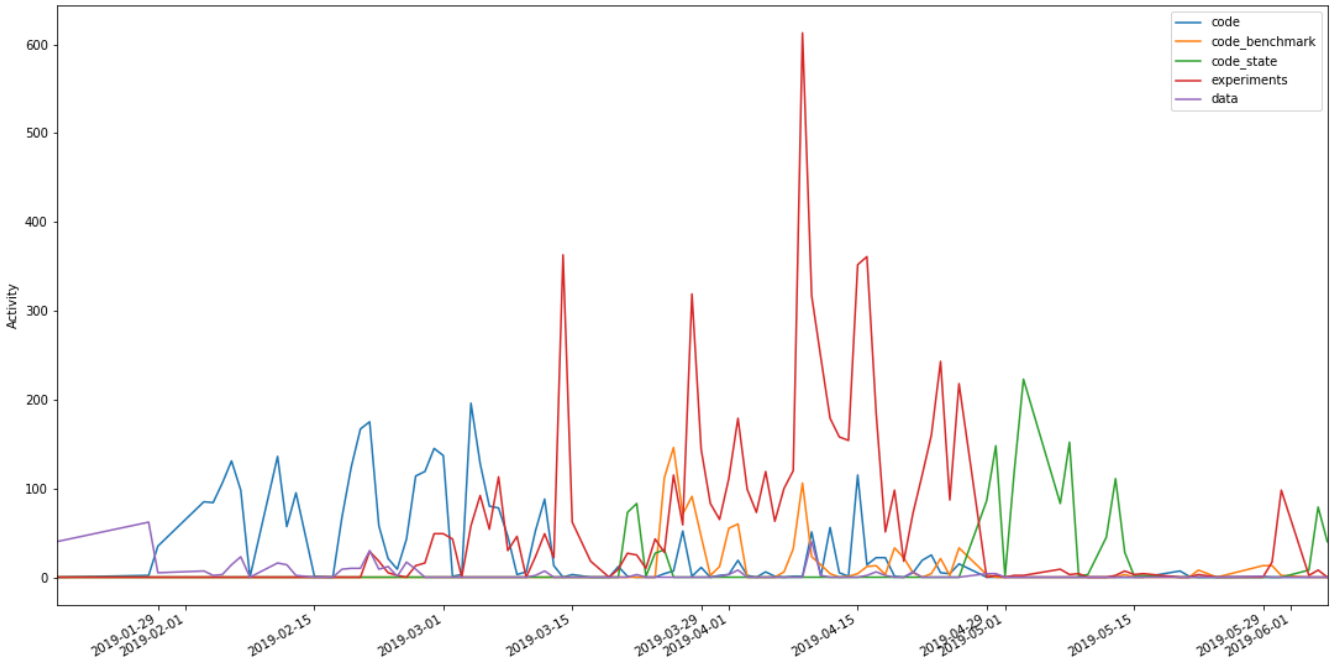


Figure 51: Logbook based on Google Drive

Figure 51 illustrates my personal activity on Google Drive. I wrote a small program that interacts with Google Drive's activity API. Within this program, I make requests to the activity within a folder and all subfolders/files. Based on the folders name, I could track my progress during my thesis. 'Activity' refers to saving files, moving files, creating new files…. In this figure, the legend represents certain parts of the platform I designed:

- Code: Main coding folder. This folder contains the code used to develop the platform, including building new models, evaluating models, pre-processing, visualisations, scheduling experiments, writing data to HHD, communication with different servers, …
- Data: This folder contains all data generated describing the road-networks, statistics, raw GPS-traces, map-matched trajectories, ….
- Experiments: This file contains all experiment data; in total, there are 22 experiments conducted. The main code controls the structure of this experiments map.
- Code-benchmark: This is a clone of 'Code' altered to testing the benchmark models.
- Code-state: This file is also cloned from the 'Code' file. This clone was altered to allow for network-wide predictions, time-autoencoding, decoding, …

During the first weeks of the thesis, the data folder grew slowly. Within this period, the map-matched data was aggregated, which takes about one week but only needs to be done once. The TRANSFOR competition ended at mid-January.

After the competition, I started revising all code and restarting the process from scratch. First, the data was regenerated. This is followed by alterations in the main code, which controls the testing platform. Some small experiments are conducted in this period to evaluate the new code. The first key event is on 15/03/2019, from this point, the platform was stable and all models were tested again but now more extensively.

Next, the benchmark code was created and preliminary experiments with network-wide predictions. The following period around 11/04 – 26/04, which corresponds to Easter holidays, was used to do the most extensive testing. During this period the servers at the company were mostly unused, so I could perform more experiments faster.

Analysing the results obtained from the refined and optimised models on 1/05. The network-wide predictions were designed. The testing of network-wide predictions results in fewer files and thus less activity. The network-wide predictions only use one file to store their information compared to 12 for previous models.

9.3 Software listing

Table 14 contains the platform is configuration and main libraries used during development.

Code	Anaconda Python 3.6
IDE	Spyder 3.0
Deep learning	Keras – Tensorflow-GPU
Network creation	Netwokx
Data structures	Pandas, Numpy
Save format	Pickle, h5, CSV
Visualisation	Matplotlib, Tikz

Table 14: Code configuration with main libraries used

The computational resources available are described in Table 15.

Graphics card	NVIDIA GeFroceTM TX 1080TI
OS	Ubuntu 16.04
CPU	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz
Memory	38 GB
Cuda	9.0
CuDNN	7

Table 15: Computational resources available

The platform allocates 267 GB, with 193 GB being allocated to experiments and saved models. The Code is spread across 200 files in different programming languages such as Java, Python and R. Because of this complex structure the code is not attached to this thesis. If any reader is interested in the code or some specific parts, you can contact me on my email[9].

[9] Toon.bogaerts@opendeusto.es